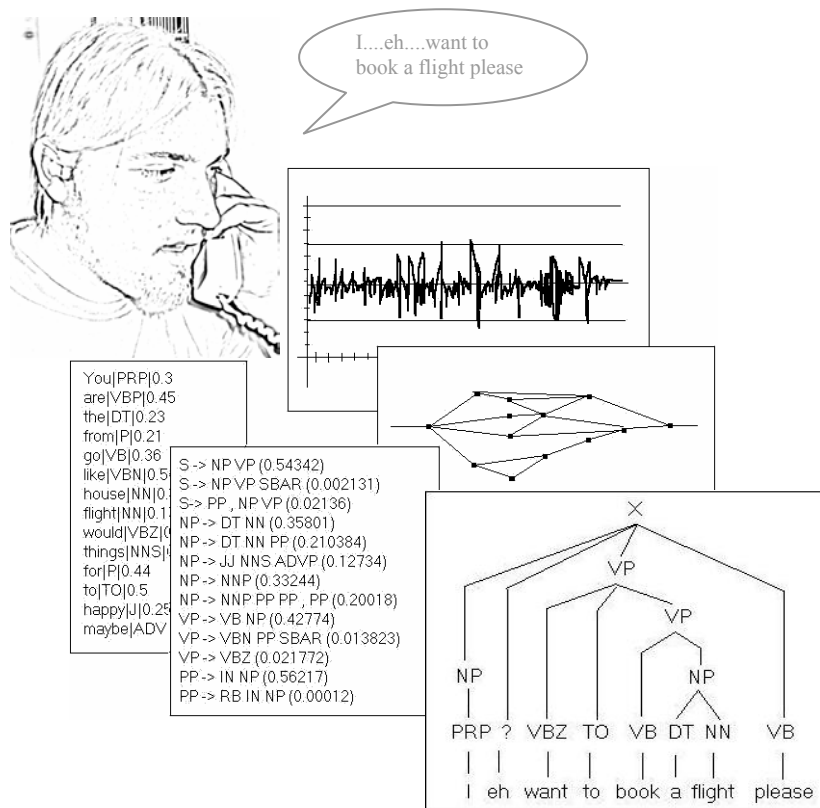


## Two Approaches to Robust Stochastic Parsing



Master's Thesis in Computational Linguistics

Technical Report No: IC/2004/97

Marita Ailomaa

Tutors:

Martin Rajman, EPFL

Jean-Cédric Chappelier, EPFL

Torbjörn Lager, GU

Lausanne, September 2004

## **Acknowledgements**

I would like to thank my tutors Martin Rajman and Jean-Cédric Chappelier for offering me this project and for guiding me through my first experience of research. I would also like to thank Vladimir Kadlec for his cooperation with the syntactic tools and Jens Ingensand for being such a good support whenever I needed it.

Marita Ailomaa

2004-09-28

## **Abstract**

NLP applications in all domains require more than a formal grammar to process the input in a practical way, because natural language contains phenomena that a formal grammar is usually not able to describe. Such phenomena are typically disfluencies and extra-grammaticality. Some robust technique is needed to deal with them. An important issue in the development of robust parsing techniques is the choice of flexibility. What precise phenomena outside the systems grammar shall the parser be able to handle? Another question is how to select the correct analysis among the great number of solutions, which are produced as a consequence of the flexibility. This report presents experiments done with two different techniques. One is based on the combination of partial parses, the other on controlled relaxation of grammar rules. In both techniques the selection of the “best” analysis is done with a statistically based ranking procedure. The grammars and test sentences are extracted from two treebanks, ATIS and Susanne. Experimental results show that the first technique has the advantage of full coverage, while the other has a better accuracy. The best performance is achieved by parsing in three passes, first with the initial grammar, then with the rule-relaxation approach and finally, if still no analysis was found, with combination of partial analyses.

## Table of contents

<b>1 Introduction.....</b>	<b>5</b>
<b>2 Robust parsing.....</b>	<b>6</b>
2.1 Applications that require robustness.....	6
2.2 Levels of robustness .....	7
2.3 Goals.....	7
2.4 Problems with different techniques .....	8
2.5 Comparing techniques .....	9
<b>3 Previous work.....</b>	<b>11</b>
3.1 Robust techniques for language disfluencies .....	11
3.2 Other robust techniques .....	14
<b>4 Technique 1: Robust stochastic parsing based on the selection of the most probable optimal coverage .....</b>	<b>20</b>
4.1 Stochastic parsing.....	20
4.2 Production of artificial full trees.....	21
4.3 Tools developed for the production of artificial trees .....	23
4.4 Test examples with the tool .....	24
4.5 Problems with this approach.....	26
<b>5 Technique 2: A robust strategy based on holes .....</b>	<b>27</b>
5.1 Introduction .....	27
5.2 The concept of holes .....	29
5.3 Specifying elements that contribute to holes.....	32
5.4 A controlled grammar relaxation strategy based on linguistic knowledge.....	35
5.5 Assigning probabilities to holes.....	42
<b>6 Validation .....</b>	<b>43</b>
6.1 Implementation .....	43
6.2 Tools .....	48
6.3 Corpora.....	51
6.4 Methodology .....	52
6.5 Some comments about the methodology .....	59
<b>7 Experimental Results .....</b>	<b>61</b>
7.1 Results .....	61
7.2 Analysis .....	62
7.3 Further experiments .....	67
<b>8 Conclusions and Future Work .....</b>	<b>69</b>
<b>9 References .....</b>	<b>71</b>
<b>Appendix.....</b>	<b>74</b>

## Table of figures and tables

Fig 1. A typical self-repair .....	12
Fig 2. Word graph for the utterance “Zondag vier februari” (Sunday February forth). .....	17
Fig 4. An artificially full tree constructed with gluing rules.....	22
Fig 5. A forest of partial parses.....	22
Fig 6. The most probable optimal maximum coverage for sentence .....	24
Fig 7. Another example of the selection of the most probable optimal .....	24
coverage and the original analysis.....	24
Fig 8. An example where the selection of the most probable optimal coverage.....	25
and the original analysis have many differences.....	25
Fig 9. Example of a sentence that doesn't parse. ....	30
Fig 10. A tree with a hole that replaces a missing rule.....	31
Fig 11. Example of rule relaxation. ....	34
Fig 12. A model of how a typical NP, VP and S are constructed.....	36
Fig 13. Three sentences where the S-rule has additional elements (besides the core). ....	38
Fig 14. The difference between the two corpora ATIS and Susanne regarding VPs.....	39
Fig 15. NP with a “filler” on the left side of the core. ....	41
Fig 16. How conjunctions are represented in Susanne and in ATIS.....	44
Fig 17. Example of relaxing a rule and allowing optional fillers at any position .....	45
Fig 18. Alternative way of relaxing grammar rules. ....	45
Fig 19. The output of the parsing option “coverage” viewed with the tool “voir_arbres”.....	50
Fig 20. The output of the tool “tree_with_holes”. ....	51
Fig 21. Examples of a correct optimal coverage, an acceptable one and a bad one. ....	55
Fig 22. Example of a correctly constructed tree with hole, an acceptable one and a bad one. ...	57
Fig 23. Three analyses an ambiguous NP .....	64
Fig 24. A bad analysis caused by the ambiguity in the grammar extracted from Susanne. ....	65
Fig 25. The architecture of a parser that analyses in three steps. ....	67
 Table 1: Disfluency classification.....	 12
Table 2. Performance rates for detection and correction of disfluencies .....	13
Table 3. Statistics on the types of rules that are missing. ....	33
Table 4. Statistics on the number of non-terminals that differs a missing rule (for an extra- grammatical sentence) from one that is in the grammar. ....	34
Table 5. An example of the classification of left-, right and core elements of NP-rules. ....	40
Table 6. Example of the classification of left-, right- and core elements of a grammar. ....	44
Table 7. Tools and parsing options in the SLPtoolkit.....	48
Table 8. Some characteristics of the two corpora used for the experiments. ....	52
Table 9. The proportions of learning and test set for the ATIS and Susanne corpus. ....	53
Table 10. Experimental results. Percentage of correct, useful and bad analyses with the two robust techniques, tested on ATIS and Susanne. ....	61
Table 11. Comparison of the two techniques. ....	61

# 1 Introduction

In NLP applications formal grammars are used for the purpose of describing well-formed sentences in natural language. But natural language is not always well formed, and even if it is, there are unlimited possibilities of how to construct grammatically correct sentences. NLP applications that rely completely on such grammars (describing a subset of the language) cannot be practically used in a large scale, because when the parser fails to derive a full analysis, it will not return any information at all.

There is of course a lot of valuable information in a sentence even if the formal grammar does not provide a full analysis. An intelligent system should be able to use that information. The user expects the system to reason in a similar way that a human would do when receiving a request that it doesn't fully understand

One solution to the problem is to make the parser more flexible, to be able to handle input outside the grammar's coverage. This report presents two directions in robust parsing. One is the approach of combining maximal partial analyses and the other is relaxing the rules in the initial grammar. Both approaches were developed at The Swiss Federal Institute of Technology in Lausanne (EPFL) using the SLP toolkit, a library of language processing tools implemented at EPFL.

This report has six main parts. In the first part the concept of robust parsing and the problems connected to it are introduced. Next, examples of some robust techniques that have been implemented in previous research are presented. In part 4 and 5 I describe the two approaches to robust parsing that are developed and tested in this project. Part 6 describes in detail the implementation, tools, data and methodology that are chosen for the experiments. Part 7 presents the experimental results. In the last part conclusions are made about the two techniques and some suggestions are given for further research.

## **2 Robust parsing**

This section briefly describes the different problematics which are connected to robust parsing. The problems involve the type of application domain, depth of analysis, general goals of robust parsing, advantages and disadvantages with different approaches and how to compare results reported by different researchers.

### ***2.1 Applications that require robustness***

A common use of natural language processing is within systems that use speech recognition, for example dialogue systems for customer support and conversational interfaces for databases. Other large fields are information retrieval and machine translation. In each field there are unpredictable natural language phenomena that will fail to be analysed with a formal grammar. The phenomena that need to be handled robustly depend on which type of application is in question. In dialogue system the system has to face spoken language, which typically contains disfluencies such as hesitations, repetitions and speech repairs.

Systems that process large amounts of written text are confronted with another problem, the variety of grammatically correct sentences that are not covered by the systems grammar.

Whether the cause of the parse failure is ungrammaticality or extra-grammaticality, some strategy is needed to handle these cases. The parser is robust if it is able to produce some analysis for all of the input. It doesn't have to be entirely correct, but it has to be useful so that the system can process it further, for example with a semantic interpreter.

## **2.2 Levels of robustness**

Research groups worldwide have developed systems with robust parsers and their approaches differ in several aspects. The notion of ungrammaticality vs. extra-grammaticality is clearly distinguished by some [2,13,18], while others attempt to solve both of them in a unified manner [14,15,19]. The approaches also differ in depth of analysis. There is lexical parsing, which means identifying syntactic features of individual words. Then there is shallow parsing, where the task is to find smaller phrases such as NPs or to form hierarchical syntactic structures without exploiting sub-categorization. Full parsers analyze with more depth. They deal with unbounded dependencies, such as the recovery of predicate-argument structure.

The depth of the analysis depends on the application domain. Machine translation, for example, requires deep syntactic analysis whereas shallow parsing is usually more adequate in information retrieval. The more complex the analysis is, the more it costs in temporal computational efficiency. The balancing between linguistic detail and computational efficiency is a central issue in the development of practical NLP applications.

## **2.3 Goals**

Some argue that the best approach to robustness is to take into account the application domain, i.e. that one technique can not be successful in all contexts [15]. But many others agree that the goal of the research in robust parsing is to find a strategy that is both domain-independent and practical. Such a parser should return the correct or a useful ‘close’ analysis for 90% or more of input sentences [4]. No strategy has been found yet that fills the criteria. In order to achieve this high number, the system has not only to solve the problem of dealing with cases outside the systems lexical or syntactic coverage (undergeneration), but also [4]:



- To provide appropriate segmentation of the input into syntactically parseable units
- To deal with disambiguation, or in other words, selecting the unique semantically and pragmatically correct analysis from a potentially large number of syntactically legitimate ones

The focus of this report is on the first point, undergeneration, so I will not describe how the other two problems have been solved, but it is important to mention them, because they have a considerable impact on the experimental results of any robust technique.

## ***2.4 Problems with different techniques***

The robust parsing problem has been approached from several directions. Statistical and connectionist approaches have the advantage that they are inherently robust. They consider all possible analyses of a sentence given an (initially) weak grammar, and they can be trained automatically from labelled corpora. But they are less capable of performing deep and detailed analysis [17].

In some approaches, semantic information has been used as a compensation for the lack of grammaticality. Early approaches were based on hand-coded grammar specific heuristics for selecting a subset of analyses and extending them at the processing of every new input word [17]. Unfortunately such rules have to be re-written for every new grammar. In general, strategies that depend on detailed semantic information tend to be domain-specific and suffer from undergeneration [4].

More recent approaches to robust parsing are based on shallow or partial parsing techniques [14,19]. Instead of constructing an analysis of the whole sentence, they attempt to analyze the largest possible fractions of the input. These types of techniques solve the computational expense problem, but in terms of accuracy they cannot compete with all other techniques [17].

The central problem that needs to be solved, independently of approach, is how to find the right balance between robustness, accuracy and efficiency. The answer seems to lie in the hybrid approach (the right combination of techniques), and this is what researchers are currently experimenting with [1,17].

## ***2.5 Comparing techniques***

One research field in robust parsing is to find methods for comparing different techniques. In most experiments the language data is taken from one or several corpora but different researchers don't use the same ones. There are many kinds of corpora at hand today, with variations in size of vocabulary and homogeneity of the text. For example a corpus with task specific dialogues between human and computer contains rather simple sentences, while a corpus with dialogues between two persons talking about an arbitrary subject may contain all kinds of interesting and unpredictable speech phenomena [16]. Written text from novels or newspapers, on the other hand, differs strongly from spoken language.

One popular corpus covering both transcribed spoken language and written text is the Penn Treebank. There are a number of files collected from different domains such as IBM computer manuals, Wall Street Journal articles and transcribed telephone conversations [12]. ATIS is a frequently used file, because the domain is task oriented human-computer interaction and this is highly relevant for experiments with dialogue systems [12]. Susanne is an example of a corpus of written text taken from mixed sources [22].

Although many researchers report results that are based on the same corpora, it is problematic to compare parser performances because of the way in which the data has been prepared for the tests. The main questions are: How large was the proportion of the corpus that was used for learning and how large was the test set? And how was the accuracy of the resulting analyses evaluated? There are some generally known schemas such as exact match, correctness of rule

application and a popular method called PARSEVAL [4] but they are not used in a standard fashion.

Now I have shown the different dimensions that need to be taken into account in solving the robustness problem for natural language. In the next chapter I will introduce some robust parsing techniques that have been developed in previous research. I've selected some approaches that are relevant to the approaches taken in this project, but it should be pointed out that these are only a subset of a great number of techniques. Some are based on completely different types of grammars and sources of information than the ones that I will present.

### 3 Previous work

In this chapter I will introduce some robust parsing techniques that have been developed in previous research. I will start by presenting some work that has been done specifically for dealing with language disfluencies. Although language disfluencies are not the main focus of this project, they are an important issue when it comes to any spoken language application and it is interesting to know what can be done with them.

The second section describes other robust techniques, which deal with both ungrammaticality and extra-grammaticality. The techniques are all shallow in the sense that they are based on partial analyses, but they vary in implementation and they are developed for different domains.

#### ***3.1 Robust techniques for language disfluencies***

As I mentioned previously, there is a difference between ungrammaticality and extra-grammaticality. The first term means that some part of the input was not covered by the grammar because the language was not syntactically acceptable, for example:

I think we need to...uh...to think about it

The second one means that the sentence was grammatically well-formed but was not included in the grammar, for example because the word order was unusual:

This, for the liberals I know, would be an understatement

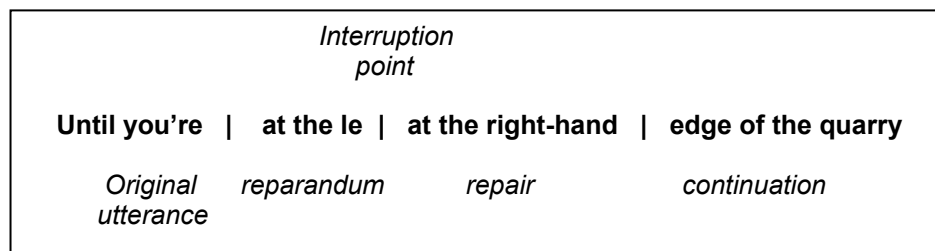
The usual manner of dealing with disfluencies is to automatically detect them and to correct the sentence by deleting words that prevent the analyzing system from deciding the true meaning of the utterance. In some approaches the detection

and correction is done before the parsing process [2, 8,18], in others it is done along with the parsing [13]. The basic idea (behind all approaches) is that language disfluencies follow certain patterns and that these can be identified by observing the word sequence and, in some cases, syntactic categories. A very detailed classification of disfluency patterns was proposed by Shriberg (1996). (See table 1).

Disfluency class	Example
filled pause	she. <del>uh</del> liked it
repetition	<del>she</del> . she liked it
substitution	<del>she</del> . he liked it
insertion	she <del>liked</del> . really liked it
deletion	<del>it was very</del> . she liked it
speech error	<del>sh</del> le. she liked it

*Table 1: Disfluency classification*

A disfluent sentence is divided into specific parts. First there is the start of the utterance. At some point it is interrupted by some unintended words. At the end of those unintended words (“edited words” or reparamdum) is the interruption point. This point is important to decide because it is followed by the words that repair the error. After the repair part is the continuation of the original utterance. Figure 1 demonstrates the segmentation of a disfluent sentence.



*Fig 1. A typical self-repair*

When all the different parts of the disfluency have been identified, the correction of the sentence is quite simple. The unintended words are removed and the sentence becomes fluent. The difficulty lies in determining the different parts of

the disfluency automatically. Some sentences may match a pattern without being disfluent:

We **had had** some real problems.

Some may match several patterns, of which only one is the correct one. For example the pattern “FLIGHT <word> FLIGHT” matches three different patterns (the annotation with M1 and R1 is described in Bear et al.(1992)):

Show the **flight** | earliest **flight**

M1                      M1

Show the delta **flight** | united **flight**

R1    M1            R1    M1

Show the **flight** time | **flight** date

M1    R1            M1    R1

Experiments that have been done by different researcher show varying results. A comparison of two approaches is presented in table 2.

	Detection	Correction
Multiple knowledge sources <i>Bear, Dowding, Shriberg (1992)</i>		
Recall	76%	43%
Precision	62%	50%
Pure Pattern Matching <i>Heeman and Allen (1994)</i>		
Recall	83%	80%
Precision	89%	86%

*Table 2. Performance rates for detection and correction of disfluencies*

The numbers indicate that the second method is more successful, but the problem of comparing performance rates is the same as I mentioned in chapter 2. The tests were done on different corpora and the classification of disfluencies was not identical. But what is common between these methods of dealing with

ungrammaticality is that the disfluencies are detected before or during parsing, but not after. This leads to some failures, when sentences are classified as disfluent although they are not. Another aspect to consider is the type of input that was used. In these tests the input was human-transliterations of speech. But since language disfluencies are a central problem in speech recognition, the techniques should be applicable on speech recognizer output (Word Hypothesis Graphs), which introduces an additional level of uncertainty and complexity [19]. This problem has not been addressed in any of the two approaches mentioned above.

Other robust techniques, which usually don't make a difference between ungrammatical and extra-grammatical sentences, are applied only when a sentence fails to be parsed with the original grammar. Some of these techniques are presented in the next chapter.

### ***3.2 Other robust techniques***

In this section I will present some robust techniques for dealing with both ungrammatical and extra-grammatical input. In the first part, I describe shallow parsing with finite-state transducers, which, at the present state of research, is the most effective general-purpose technique.

The second part describes several techniques based on combination of partial parses. The underlying idea behind these techniques is to use the partial information that is available when a full parse fails, and to put the pieces together in a meaningful way. It may seem to be a simple task, but very few of the partial analyses that are produced are usually good candidates for a repair. They have to be selected with care, using different constraints and heuristics.

#### **3.3.1 Shallow finite-state parsing**

The goal in shallow finite-state parsing is not to analyze sentences but unrestricted text. The analyses are returned in form of chunks and they do not

correspond to syntactic trees, which is the usual output when parsing with grammars. The text is simply annotated with tags representing lexical categories and/or syntactic labels. In more advanced parsing they may also have some subject-predicate marking. An annotated sentence may look like [1]:

[VC [NP le president NP] / SUBJ [PP du CSA PP], [NP Jacques NP] [NP Boutet], a  
decide [VC de publier [NP la profession NP] [PP de foi PP] VC] VC] ./SENT<sup>1</sup>

The tagging procedure is done with several transducers. Each one adds specific types of tags and lets the input remain untagged if it doesn't match the right pattern. This way, the analysis can never fail completely. For example the first transducer adds NP (nominal phrase) tags for all potential subjects. Then the rest of the NPs and some other phrases such as AP (adjective phrases) and PPs (prepositional phrases) are tagged, because they are relatively easy to recognize. The order of the transducers is crucial for the robust behavior. The ones at the top of the sequence cover the most frequent linguistic phenomena. Segments that do not match the constraints of these transducers are passed on to transducers further down in the sequence that describe less frequent phenomena. When the parser has several options on how to tag, it makes some temporary tags that can be removed or made permanent later on.

The principle for the tagging process is to recognize typical starts and ends of phrases. For example an NP is likely to start with a determiner and end with a noun. Subcategorization of NP-complements is not allowed in this type of parsing, because it produces too much ambiguity.

At all levels of annotation, constraints are applied to eliminate some of the temporary tags. The constraints are mainly syntactic, for example there can only be one subject. When all the possibilities have been explored for making a syntactically correct annotation, and there are still sequences that are not annotated, some of the constraints are allowed to be violated. For example, a

---

<sup>1</sup> The CSA president, Jacques Boutet, decided to present his profession of faith.



verb chunk is expected to have a finite verb, but if none is present, then an interpretation without a finite verb is accepted.

Shallow parsing with finite-state transducers is highly accurate and efficient. The syntactic annotation that it produces is useful in many domains, for example in information retrieval. But there are some domains that require a deeper form of analysis in order to be successful, in particular applications for machine translation. In the next chapter I will present one such application together with other applications that use approaches based on partial analyses.

### **3.3.2 Combining partial analyses**

Approaches to robust parsing that are based on the combination of partial analyses can be found in different domains. In the Verbmobil project [19], which is a large-scale research project, the area of interest is automatic translation of spoken language. In such applications, the input is processed in several steps, first with a speech recognizer, then with a structural analyzer and after that with the automatic translation modules.

Another application where speech recognition is needed is dialogue systems. Van Noord et al [14] present a solution for processing spoken language input in a practical way with robust grammatical analysis instead of concept spotting.

Also systems with text-based input have used the concept of combining partial analyses to achieve robustness. Oltmans developed a robust parser which was especially designed for the Condorset indexing system [15].

These approaches follow two different directions. The ones dealing with spoken input are mainly focused on solving two problems: (1) disfluencies and other ungrammaticalities and (2) the uncertainty which comes from the preceding speech recognizing phase. In fact, the input to the parser is not a sequence of words as you would expect, but a Word Hypothesis Graphs (WHGs). A WHG is a directed graph where the nodes are points in time and the edges are possible

words (or parts of words) that the user may have uttered. Each edge has a weight representing the likelihood that the given part of speech was pronounced. An example of a word graph is given in figure 2.

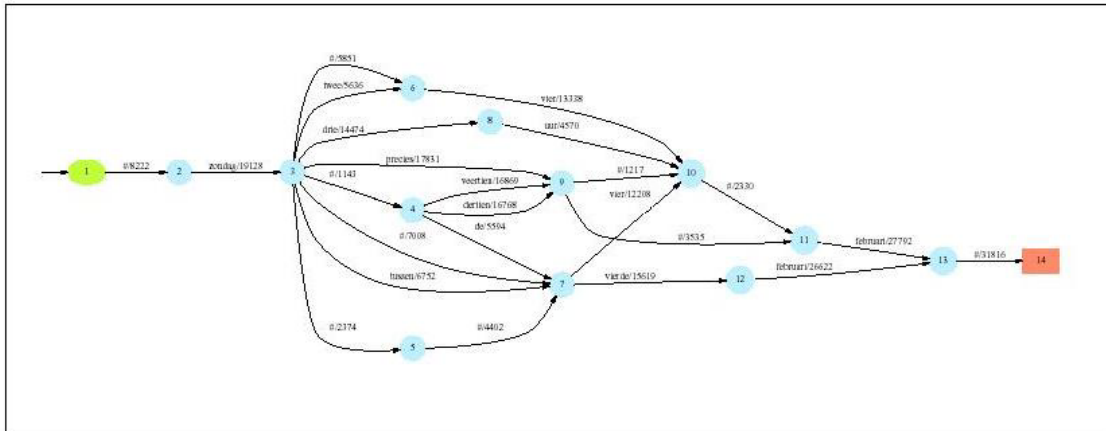


Fig 2. Word graph for the utterance "Zondag vier februari" (Sunday February forth).

In the Condorset indexing system, the problem is not language disfluencies, but extra-grammaticality. The text that is analyzed by the parser is structurally complex and rich on important information. The structural analysis is a necessary middle step in the process to generate the right index concepts that are delivered in "linguistic packages". For example, consider the highly extra-grammatical sentence:

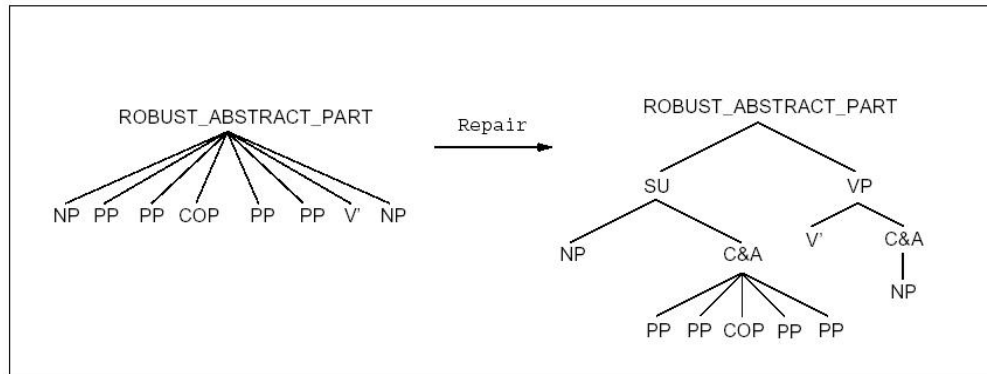
It was found 1,2-benzisoxazole-3-methanesulfonamide was extremely effective in a total 29 cases of posttraumatic (of mild to moderate) epilepsy (41% of patients).

The grammar is likely to fail to construct a full syntactic analysis for this sentence, but a robust parser should be able to perform some level of analysis, so that the indexing process afterwards is able to give the following linguistic package:

was\_effective (1,2- benzisoxazole-3-methanesulfonamide, posttraumatic epilepsy)

The methods for making the parser more robust are, in all three applications, based on combining partial analyses. First an attempt is made to construct a full analysis and when that fails, a robust component finishes the parsing. In both applications where the input are WHGs, the strategy is to find the optimal path in the word graph, when the initial parsing step has annotated the edges with syntactic categories. The optimal path can be decided with different measures. Some of the criteria that the two approaches have in common are the “acoustic score” introduced by the speech recognizer, the smallest possible number of categories and a small number of skips. But there are some differences that are due to the application domain. In the automatic language translation case, some partial analyses are combined with general and language specific rules. The new edges produced with these rules have a “confidence value”, which pushes up the score of these edges. In the dialogue system example, one measure for selecting the optimal path is to decide how well the contents of the partial analyses in that path correspond to the dialogue context. For example if the last question posed by the system requires an answer containing the name of a city, then an analysis where this can be found has a better score than some others that don’t.

Oltmans approach to robustness is based on grammars. When parsing with the core grammar fails, a peripheral grammar produces an analysis consisting of chunks, which are treated further in the robust component. He makes a distinction between major chunks, such as NPs and VPs, and minor chunks, like adverbs and commas. The robust component performs three tasks. It removes messy details (some minor chunks) that “block” the full analysis, and then it repairs the syntactic structure using specific rules. Finally it produces a canonical format, from which the indexing concepts can be produced. Each of these three steps is done with a separate grammar. An example of the repair step is given in figure 3.



*Fig 3. Example of repair transformation. These operations transform flat structures of chunks into a format that is nearly canonical. The tree structure to the right is now prepared for the generation of the canonical format.*

The developers of the three techniques report positive results concerning both accuracy and computational efficiency. But one of the goals of Van Noord et al, when performing structural analysis as an alternative to concept spotting, was to investigate if it would lead to better speech recognition accuracy. No evidence was found for that.

The main problem with these techniques is that they depend on elements that may not be scalable for general-purpose approaches. The grammars that Oltmans developed for his robust component are tailored for constructing an output suitable for the Condorset indexing system, but these grammars were hand-coded and may not be adaptable in all contexts. Also in the Verbmobil project, the processing of partial analyses involves the writing of special rules that cover some semantic or language specific phenomena.

The next section describes an alternative approach for selecting partial analyses that is not dependent from any semantic, language specific or domain-specific rules. This technique represents one of the two approaches to robust stochastic parsing that are tested in the project.

## **4 Technique 1: Robust stochastic parsing based on the selection of the most probable optimal coverage**

In this section, I will describe one of two approaches to robust parsing, which are the subject of my research. It is a robust technique based on combining partial analyses in a stochastic parsing context. The technique is in many aspects simpler than the ones described in the previous section, but it has another advantage, which is that the parsing is done with a probabilistic grammar. It is important to note that this technique does not guarantee correct or even useful analyses for sentences that initially failed with a formal grammar. I will demonstrate why it is so. But to begin with, I will say a few words about the concept of stochastic parsing and probabilistic grammars.

### ***4.1 Stochastic parsing***

Many parsers use formal grammars to analyze language input. Stochastic parsing has the difference that the rules in the grammar are assigned with probabilities. The parse trees that are derived by the grammar therefore have probabilistic scores. It means that, given a sentence and a stochastic grammar, the parser is able to select the most probable analysis.

Probabilistic parsing techniques have recently gained a large interest in NLP [3,11]. The grammars are trained automatically from previously annotated data. With all the treebanks that are available today, it has become much easier to test these techniques. Grammars that are extracted from treebanks tend to be large and highly ambiguous, which has driven the efforts to develop more efficient parsing algorithms [6].

Briefly, the calculation of the total score of a tree is based on the product of the probabilities of the rules that derived the tree. For example, if there is a tree with root node S, produced by a rule having the probability 0.5:

$S \rightarrow NP VP (0.5)$

Then the score of that tree is calculated with the following formula:

$P(S) = 0.5 * P(NP) * P(VP)$  where  $P(NP)$  (resp.  $P(VP)$ ) is the probability of the subtrees dominated by NP (resp. VP).

An important feature of a probabilistic grammar is that all the rules having the same left hand side, for example all the NP-rules, have to have probabilities that together sum up to 1. I will return to this issue further down.

## **4.2 Production of artificial full trees**

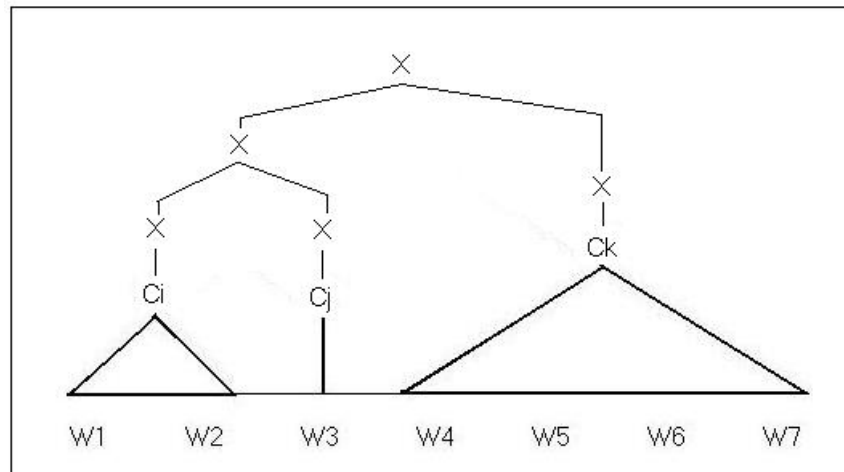
In this chapter I describe the first robust technique that is subject of my research. The technique is based on selecting partial analyses and producing an artificially full tree.

The partial analyses that are provided when a full parse is not possible are stored in a forest of partial parses. This is a method that we have already seen in other applications [14,19]. From those partial analyses, simply combining them into one full tree can produce an artificial tree. This has also been done before, as we saw in section 3. Oltmans uses a technique where he removes some chunks that “block” a full analysis and then tries to fit the remaining chunks into a canonical format. The alternative approach that is taken in this case is to select the sequence of most *probable* partial analyses and to use simple gluing rules to connect them into one full tree. The rules may have the following form:

$X \rightarrow X X$

$X \rightarrow C_i$  where  $C_i$  is the  $i$ -th non-terminal in the grammar

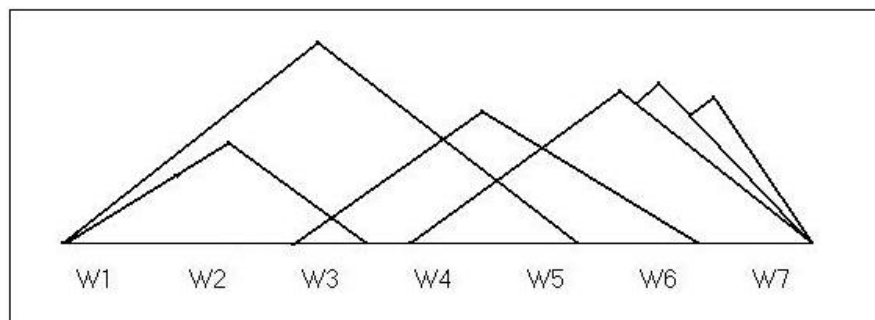
An example of a glued tree is shown in figure 4.



*Fig 4. An artificially full tree constructed with gluing rules.*

The purpose of this robust technique is not only to select the best sequence of partial analyses when the initial grammar fails, but to be able to derive a correct full tree. The gluing itself does not produce a correct full tree. It only selects the parts which are possible fragments of a correct tree. This is how most of the previous approaches have done. But a further action is required to repair the artificial tree and to produce the correct full tree.

As I mentioned earlier, there can be many partial analyses in the forest that are not good candidates for a repair. For example there can be many different solutions for one specific sequence of words and there can be overlaps between trees.



*Fig 5. A forest of partial parses*

An assumption is now made that the selection of the optimal set of partial analyses is the key to a successful derivation of the correct full tree. The gluing of partial analyses is considered optimal if the sequence of partial analyses is non-overlapping, spans the *whole* input and has the maximal score. But this is not sufficient. If the scoring is only based on the individual scores of the partial analyses, a gluing with pre-terminals (lexical trees) will always be ranked higher than one with deeper structures. This is evident from the fact that the individual score of each tree is lower than or equal to 1. A larger tree consisting of sub trees has consequently a lower score than each of its parts.

In order to obtain artificial trees that have maximal partial analyses, the attention has to be drawn to the scoring of the gluing rules. The lower the probabilities assigned to the X-rules are, the fewer partial analyses will be glued.

Experiments show that if the gluing rules are assigned a probability strictly lower than the lowest probability of any rule in the grammar, a glued tree with the smallest number of trees will be ranked highest. When there are multiple solutions that have the same number of trees, they are ranked by the individual scores of the partial analyses.

### ***4.3 Tools developed for the production of artificial trees***

In cooperation with another natural language project [20] a tool was developed for selecting the most probable optimal maximum coverage from a forest of partial trees. *Optimal maximum coverage* stands for the set of non-overlapping partial analyses that spans the whole input and that consists of maximum trees. That is if there is a tree  $T$  in the maximum coverage and there exists a tree  $T'$  such that  $T$  is a sub tree of  $T'$ , then  $T$  is equal to  $T'$  [20]. This tool is integrated in SLPtoolkit, an NLP-tool developed at EPFL [21].



#### 4.4 Test examples with the tool

The typical pattern among the sentences that have been tested with the new tool is that the partial analyses in the most probable optimal coverage are of two types: wide trees that cover nearly the whole input and small trees spanning only one or two words. An example is demonstrated in figure 6.

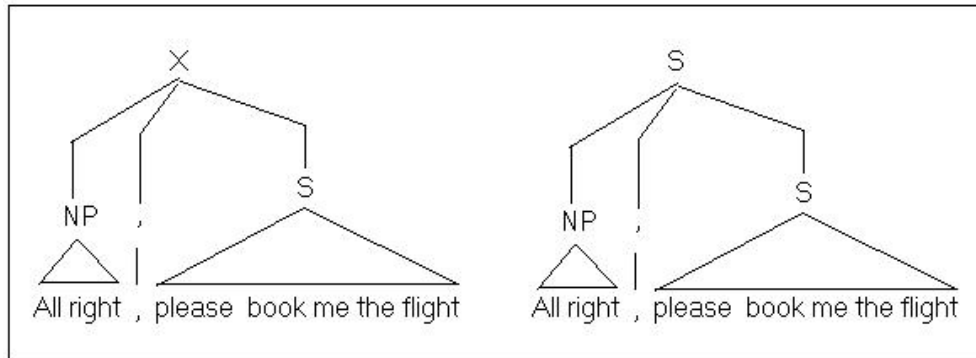


Fig 6. The most probable optimal maximum coverage for sentence "All right, please book me the flight" To the right is the original analysis.

In this example a wide tree follows two partial analyses of length 2 and 1. The selection is indeed optimal. All that needs to be done to obtain the correct full tree is to add a new S-rule to the grammar. Another example, which also doesn't seem too hard to repair, is shown in figure 7. In this example the partial analyses consist of one wide tree followed by two small trees of length 1.

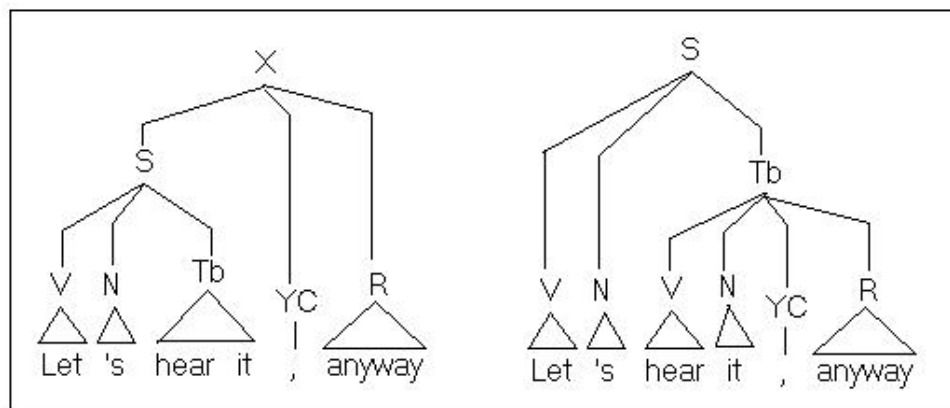
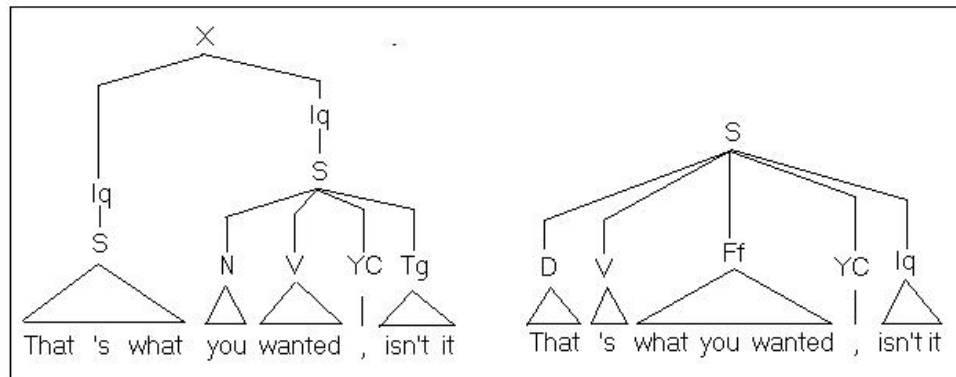


Fig 7. Another example of the selection of the most probable optimal coverage and the original analysis

The difference between this example and the last one is that a new rule for the top node is not sufficient to arrive at the correct full tree. There is only one rule missing though, and with some linguistic information, it should be possible to decide how the partial analyses should be connected.

But let us look at an example that is not so easy to repair:



*Fig 8. An example where the selection of the most probable optimal coverage and the original analysis have many differences*

As can be seen from figure 8, the selection of partial trees corresponds to the criteria for an optimal coverage, but the difference between this set of partial analyses and the correct full tree is considerable. There is no evident way of repairing this structure so that a correct full tree will be produced.

Some other possible difficulties may arise when trying to recover the correct structure from the glued tree of partial analyses. For example if the optimal coverage consists of many trees, then it becomes a more complex task to make the right decision about when to generate a new mother node for some of the partial analyses and when to subcategorize a tree under another.

In the next section I will discuss the central problem with this approach and motivate why I've chosen to test a different approach, which I will describe in section 5.

#### ***4.5 Problems with this approach***

The gluing approach described in this section is based on a scoring function that selects partial analyses with heuristics. As we have seen, the chosen heuristics do not take into account any linguistic knowledge and therefore do not guarantee that the partial analyses are really the “best” ones for gluing. Some linguistic knowledge would possibly prevent the selection of partial analyses that cannot lead to a correct full parse.

One way of going about the problem is to not attempt to parse as much as possible, but to parse smaller fragments that are more likely to be correct and to add more and more structure with other methods. The solution that was chosen in the Verbmobil project was to extend the heuristics with semantic and language specific rules that describe how partial analyses can be combined. For example a combination of a verb with an appropriate complement results in a partial analysis that has a high *confidence value*. If there are several partial analyses competing for the same sequence of words, the one with a high confidence value will be preferred before the others.

The disadvantage of such approaches has already been pointed out before. It demands the writing of specific rules, which will cover some (but not all) possible phenomena occurring in natural language. But the main problem with the concept of selecting partial analyses is that there is not a real clue to why the initial grammar failed to describe the input. If such information was available, it could be a help in selecting the partial analyses as well as recovering the correct full analysis. In the next section I will describe a technique that takes advantage of stochastic parsing and constructs a full analysis where the “uncovered” parts are clearly distinguished.

## **5 Technique 2: A robust strategy based on holes**

In this section I describe an approach to robust stochastic parsing that is based on the concept of holes. First I give a general description of the framework, motivating the approach by comparing with other approaches taken in the literature. Then I define the term “hole” in the context of robust parsing. After that I propose a method for narrowing down the concept to avoid overgeneration. Subchapter 5.4 describes restrictions that need to be applied in form of controlled rule relaxation to assure that the holes are produced in a meaningful way. In the last chapter I describe the aspects that involve the assignment of probabilities to the rules in the robust stochastic grammar.

### ***5.1 Introduction***

In most applications the robust component is used only when parsing fails with the initial grammar. This decision has been made because of the disadvantages that follow when treating the input robustly before or during parsing. Those disadvantages are mainly the increase of complexity and the risk that an ill-formed solution is accepted before a well-formed analysis was found [19]. In this approach, I have decided to take the same path: to perform the syntactic analysis in two steps, first with a formal grammar and then with the robust component.

The focus is on extra-grammatical phenomena. A part of the reason is that the data available for testing contains too few spoken language phenomena. The test results would not represent the behavior of the robust component. But a second reason is that I believe there is an advantage in separating extra-grammatical phenomena from ungrammatical. In this way, the parsing procedure could be performed in three steps instead of two. The robust component could for example first handle ungrammaticalities by removing the parts that disturb the fluency of the sentence and then, in a second step, try to re-parse the sentence, using the robust facilities only if the initial grammar rules still fail. This, however, is another problem of research.

A strategy for handling extra-grammatical sentences can be based on the study of real examples to see what makes them extra-grammatical. In natural language there are of course an infinite number of syntactic constructions, so it may seem impossible to try to predict all the types of grammatical constructions that may appear outside the grammar's coverage [4]. Nevertheless, many of the extra-grammatical sentences that cannot be described by the grammar are only slightly different from other grammatical sentences that are analyzed successfully by the parser. For example, take a look at the following two sentences which are both grammatical but only the first one is analyzed successfully with a grammar containing more than 1000 rules.

1. The answer is yes
2. The answer is of course yes

This example is perhaps not very convincing. Why not simply add a new rule to the grammar that takes care of this small variation? Well, if the grammar is to be extended to cover all possible variations of the same basic syntactic structure, the number of rules has to be increased remarkably, which will lead to more ambiguity and consequently less accuracy. Depending on the implementation of the parser, the execution of the structural analysis may also become much less efficient. But most important of all, whatever the efforts to extend the grammar, there always will be cases where the grammar fails to describe a sentence!

Instead, broad coverage can be obtained by using the existing rules as often as possible and to relax them when needed. This goes along with the lines of the *constructive* approach to robust parsing, which can be characterized with the following statement [5]:

*Broad coverage is achieved through the relaxed approach. Constraints described in the rules are not meant to restrict the language, but rather at identifying the most likely interpretation of any given input*

Concretely it means [5]:

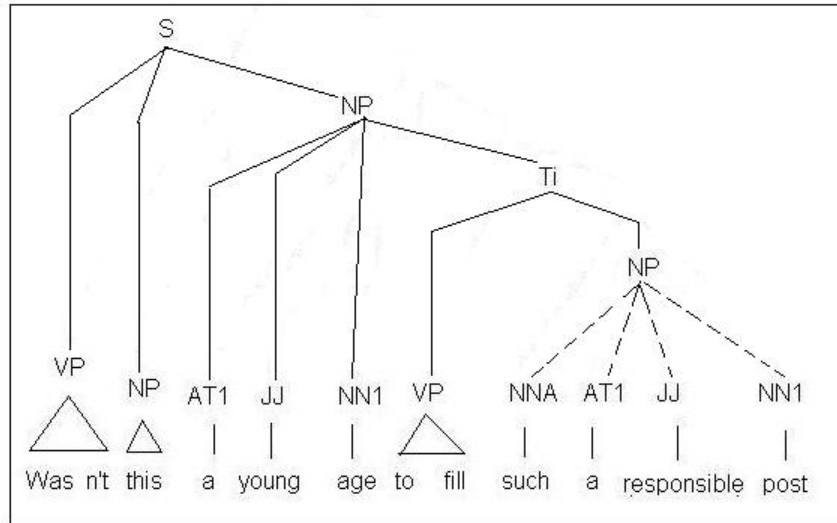
*In case of parse failure, a second pass is performed with some specific conditions being relaxed in the rules. The resulting parse keeps track of which rules and which conditions have been relaxed, so that a diagnosis for the relaxation is available.*

To keep track of where the extra-grammaticality has occurred is important in order to distinguish the parts that are described by the initial grammar and the parts that are not. It is also important to relax rules in a controlled manner. Otherwise there will be many parses that are not legitimate. With the strategy taken in this project the goal is to satisfy both conditions.

The idea is to allow “holes” in the syntactic structure. They are a kind of link between what was expected in the particular part of the structure and what was actually found. In the next chapter I will explain in more detail how it works.

## **5.2 The concept of holes**

An extra-grammatical sentence fails to be analyzed if there are one or several rules missing in the grammar. If the missing rule is near the lexical level, it may prevent a large part of the tree from being produced (with a bottom-up parser). An example of such a sentence is given in figure 9.



*Fig 9. Example of a sentence that doesn't parse. The missing rule is marked with dotted lines.*

As can be seen, there are several nodes on top of the “missing part”, which could be derived if the one missing rule was present. A way of solving this problem is to make a bridge between the partial analyses below the missing part and the node above. If this is possible, a nearly correct full tree can be produced which keeps track of the missing part. Instead of the rule there will be a hole.

The hole consists of a root that is connected to a mother node, and a number of leaves that are connected to partial analyses. The internal structure of the hole is not necessarily important. An example of what a tree looks like when it contains a hole is given in figure 10.

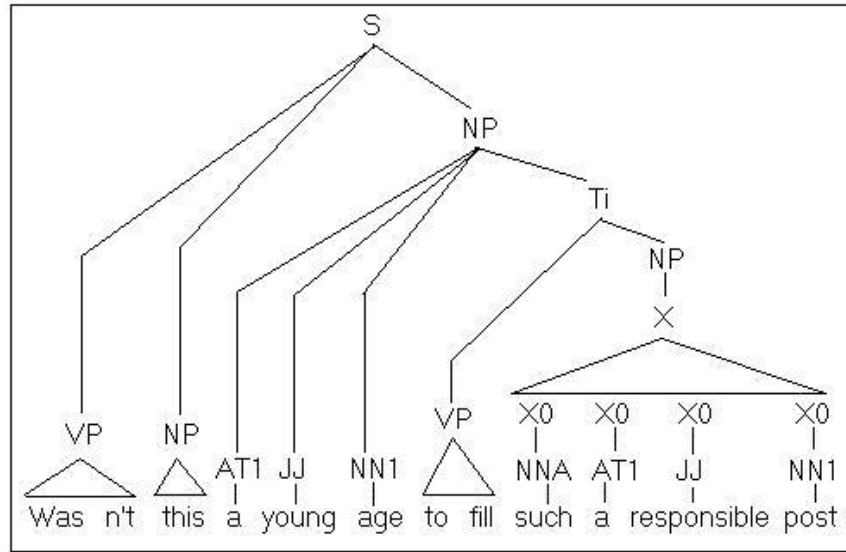


Fig 10. A tree with a hole that replaces a missing rule

In this example, the root is connected to a NP that is covered by a rule in the grammar ( $T_i \rightarrow VP\ NP$ ). The leaves are connected to the partial analyses that could not be matched to that node (NP) by any existing rule. To produce such trees with holes, some basic rules can be added to the grammar. If holes are allowed to occur in any position of the tree, the rules are:

$C_i \rightarrow X$   
 $X \rightarrow X\ X$   
 $X \rightarrow X_0$   
 $X_0 \rightarrow C_j$

$C_i$  and  $C_j$  are the  $i$ -th and  $j$ -th category represented in the grammar or lexicon. These rules are very similar to the ones that were proposed in the approach in section 4, but the first rule makes this approach different. We are not attempting to select a sequence of partial analyses that spans the whole input. On the contrary, the goal is to be able to produce a full tree with the fewest holes possible. The above rules are too general though. Observations of extra-grammatical sentences show that not all categories are likely to be the subject of holes. Nor are all the categories likely to be leaves of a hole. There is a



dependency between the root and the leaves that has to be taken into consideration. In the next chapter I will show how this can be done.

### ***5.3 Specifying elements that contribute to holes***

The problem with the concept of holes, as it is described in section 5.2, is that it doesn't make any restrictions to where the holes can appear in the syntactic structure and which partial analyses that are connected to the leaves of those holes. Some elements are always necessary to form specific structures. For example a VP has to contain a verb. So if the root of the hole is connected to a VP-node, then one of its leaves should be a verb. But usually a rule has several elements on the right hand side and only one of them is the "key" element (or "head") that makes it possible to identify the correct category of that phrase.

NP -> Det    Adj    **N**    PP  
         elem1 elem2 key elem3

Those other elements should also be taken into account in the modeling of holes to avoid illegitimate solutions.

#### **5.3.1 Roots**

Returning to the example in figure 9, we saw that a sentence may fail to be parsed because one single rule is missing. In this particular example the failure is caused by a missing rule for the category NP. It is not a coincidence. Some phrase types are more likely to be the cause of undergeneration than others. Table 3 shows the distribution of category types of missing rules in an experiment with extra-grammatical sentences in the ATIS and Susanne corpora<sup>2</sup>.

---

<sup>2</sup> The technical details of the experiment are the same as those described in section 6.4

Category of missing rule	Frequency
<b>NP</b>	<b>18</b>
<b>S</b>	<b>17</b>
<b>VP</b>	<b>11</b>
SBAR (atis)	5
L (susanne)	3
M (susanne)	2
Q (susanne)	2
Others	3

*Table 3. Statistics on the types of rules that are missing for a sample of 40 extra-grammatical sentences*

A look at the numbers shows that the missing rules for most of the extra-grammatical sentences are of category S, NP and VP. When observing the rules in a well-covered grammar, one will see that the rules that dominate are of these three types. They are the “heaviest” phrases in a syntactic structure, because they can be constructed in most various ways and be built up from many different kinds of elements. In an experiment done by [13] similar numbers were met. The context was language disfluencies, and the purpose was to find out which categories of phrases that are the most likely ones to be disfluent. Although the context is different, the similarity of the results is quite interesting, because it shows that some syntactic categories are more problematic than others. As a conclusion from this, one measure that can be used in the parsing process is to prefer holes for which the root is of one of these three kinds. For any other categories in the grammar, if forbidding them to have a hole is a too strong restriction, then one can at least say that they are very unlikely candidates.

### 5.3.2 Leaves

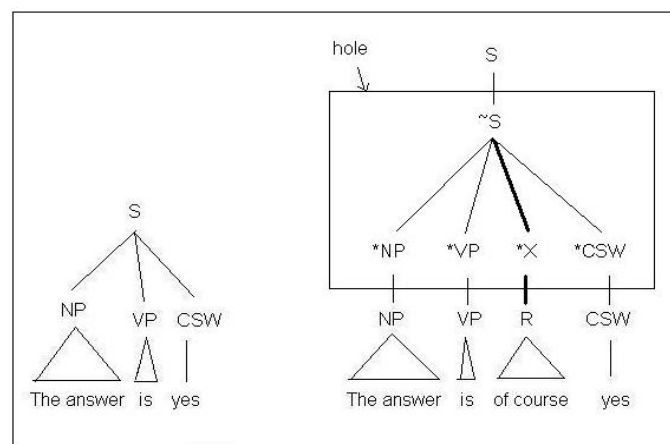
For constraining the types of leaves that may contribute to a hole, some other measures have to be taken. The idea is to take advantage of the rules that already exist in the grammar and to relax the one that best matches the sequence of partial parses that forms the extra-grammaticality. Similar methods have been tried before under the name of *Minimum Distance Parsing* [9,10] (I will

say more about them later). By doing this relaxation, the sentence can be parsed so that the hole corresponds to an “almost recognized” structure. The idea is motivated by the fact that many extra-grammatical sentences are not so different from other grammatical sentences, which are covered by the grammar. An experiment made on the Susanne and ATIS corpora shows the tendency<sup>3</sup>. The results are presented in table 4.

Difference between missing rule and rule in the grammar	Frequency
One non-terminal	43
Two or more non-terminals	3
No close match	15

*Table 4. Statistics on the number of non-terminals that differs a missing rule (for an extra-grammatical sentence) from one that is in the grammar. The numbers are taken from a sample of 40 sentences.*

To give an example, consider the sentence “The answer is of course yes”, which I mentioned earlier. The rule that is missing from the grammar to produce the correct analysis is of category S. Among the S-rules that are in the grammar, the one that parsed “The answer is yes” may be the closest one. If this rule is relaxed by allowing an extra element, the hole will be an “almost recognized” S. Figure 11 shows how the full tree would be produced.



*Fig 11. Example of rule relaxation. To the left is a successfully parsed sentence, and to the right is a sentence with a hole containing an “almost recognized” S*

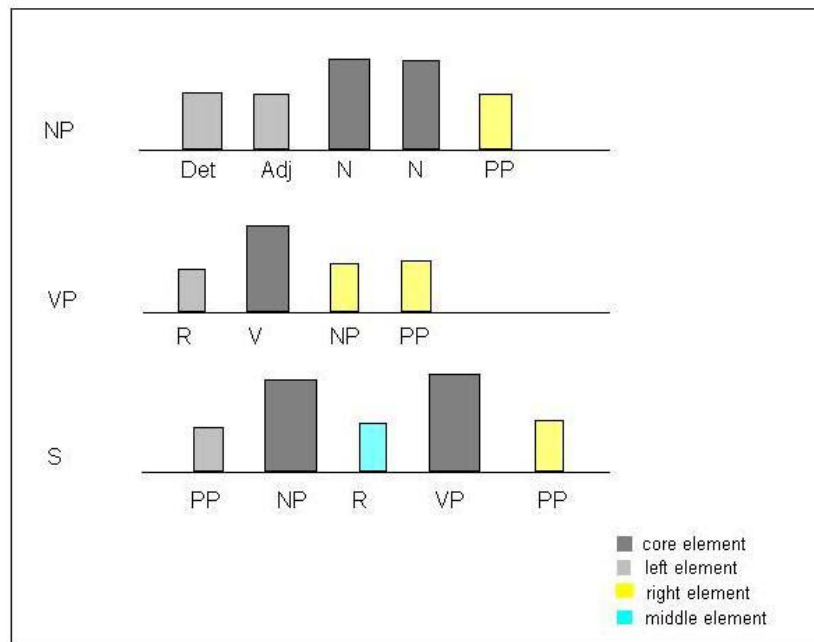
<sup>3</sup> The experiment is based on the same extra-grammatical sentences that were mentioned in section 5.3.1

In summary, the strategy that I've described is based on allowing "fillers". Fillers are some additional elements that are allowed to be absorbed in certain positions of the syntactic structure if the surrounding elements can be parsed by a rule in the initial grammar. Still, there may be too many possibilities of how an "almost recognized" structure can be produced without being a correct one. A further refinement of the method has to be made in order to achieve a higher accuracy. A way of doing this refinement is proposed next.

#### ***5.4 A controlled grammar relaxation strategy based on linguistic knowledge***

For finding a way to restrict the types of elements that may act as fillers in a grammar rule, it helps to understand how a valid phrase is usually built up. Focusing on the three types of phrases S, NP and VP, the most important thing that differentiates them are the cores, also referred to as "heads". A NP has nouns, a VP has a verb, and S has a NP and a VP. There are of course exceptions to this classification, but to make things simple, we consider only the most standard constructions.

The cores can be surrounded on both sides by other elements, and the types of those elements depend on the phrase in which they occur and on which side of the core they appear. The number of elements is not specific. They can be zero or more. A model of the three phrases is shown in figure 12.



*Fig 12. A model of how a typical NP, VP and S are constructed*

This view on how to construct phrases is similar to the idea behind shallow parsing. The strategy that Jean-Pierre Chanod describes is to identify the left-most and right-most elements of each phrase. The difference is that the elements between the left-most and the right-most element are allowed to be anything [1]. Another difference is that the goal in shallow parsing is not to parse whole sentences but unrestricted text, so there are no specific rules for the S category. Instead there are “verb chunks”. The essential thing is that in both ways a phrase is described by some typical elements occurring in it.

#### **5.4.1 Nominal Phrases (NPs)**

Starting with NPs, the usual construction is that the core elements (the nouns or pronouns) can be preceded by articles, adjectives and other words that describe the core element. Complements such as PPs and relative clauses follow after the core. Theoretically a NP can be of any length, so it is impossible to write rules that cover all possible constructions, but even though real life occurrences of NPs

are of limited length there is a large variation of constructions, which makes robustness essential in the parsing process. To exemplify this, an NP can be so short that it consists of one single word like “I” or it can be very long:

“All the non-stop flights from DFW to Stapleton International that leave before noon and are on wide-body air crafts”

In more sophisticated parsers, the robustness is not only a matter of allowing new elements to appear in the phrase. Some grammars have constraints on the grammatical agreement between words. The robustness, in such cases, means allowing agreements to be violated. To give an example, in French the determiner and head word should normally agree in gender, but there are phonological phenomena, that allow them to have different forms in specific cases, like [5]:

Mon **adorable** chat  
My [**masc**] adorable[] cat [**fem**]

By relaxing the constraints for agreement, some ungrammatical phrases may be accepted, such as:

\*Mon chatte  
My [masc] cat [fem]

This is acceptable, because robust parsers accept ungrammatical utterances as a way to increase coverage and hence robustness [5]. The important thing to keep in mind is that the sentence is initially parsed with a non-robust parser, so any spurious, unwanted analysis will be prevented if a complete analysis is available in the first pass.

### 5.4.2 Full sentences and verb phrases (S and VPs)

Sentences are perhaps the type of phrases that are the most unpredictable ones. The core has generally two parts, a NP and a VP. Additionally, there can be elements not only before or after the core but also between these two main parts. To give an example of such sentences, observe the trees in figure 13 below:

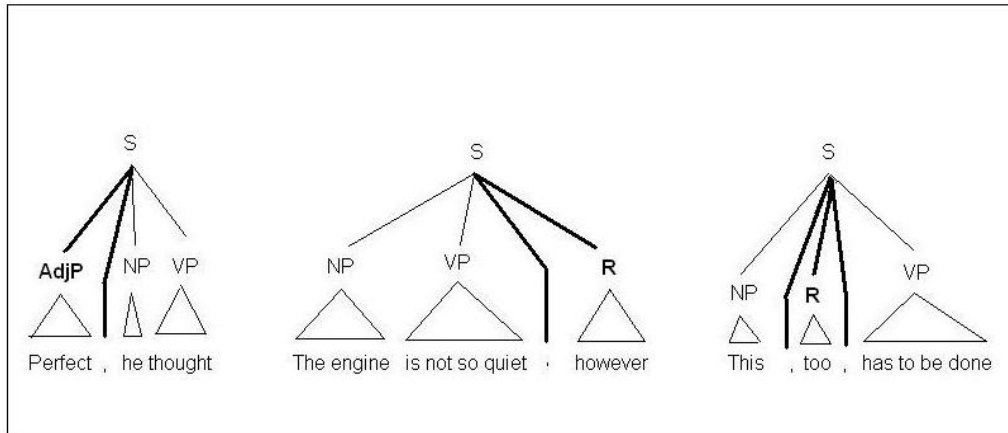


Fig 13. Three sentences where the S-rule has additional elements (besides the core).

Another source to unpredictability is that a sentence can be perfectly valid without having a core. This depends on the grammarian's interpretation of a valid sentence. In the Susanne tree bank, for example, the following rule can be found, where a VP is not required in S:

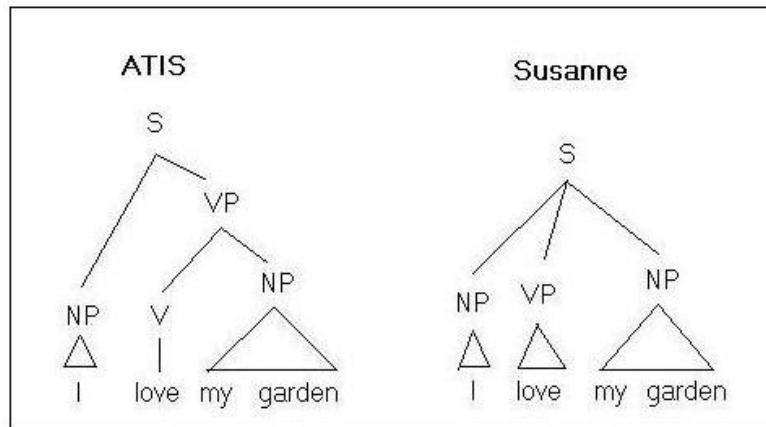
$S \rightarrow NP PP$

Verb phrases vary in a different way. First of all, there is a difference in how they are interpreted in different grammars. In the ATIS treebank a VP is basically constructed with a verb and some complements.

$VP \rightarrow VB NP PP$

In Susanne VPs consist of one or several verbs that can be preceded by some smaller words like “and” or “almost”. Other elements, like NPs and PPs, that

could be considered as verb complements are instead complements in the S category.



*Fig 14. The difference between the two corpora ATIS and Susanne regarding VPs*

The reason that I mention this difference is that I think a good strategy to rule relaxation should be independent of the interpretations made by grammarians about how a phrase is constructed. In the next part I will show how this is possible.

### **5.4.3 Relaxation rules based on the initial grammar**

From the examples in the previous sections, we can observe that the non-core elements are to some extent different for different types of phrases. For example NPs have determiners and adjectives whereas VPs don't. But there are elements that are valid both in NPs and VPs, such as PPs. This is a common source for ambiguity.

The position of the elements also plays a certain role. For example a PP can be a complement in an NP but does not usually occur on the left side of the core.

Keeping in mind that different grammars have different interpretations about how phrases are constructed, we are ready to formulate some relaxation rules that will



control the production of full trees with holes and at the same time take into account the specific features of the initial grammar in use.

The key idea is to identify, for each type of phrase, which are the valid left and right elements and to classify them according to that. The information is retrieved from the existing rules in the grammar. This can be done automatically, if the core elements can be recognized first. In cases where a core is not present in a grammar rule, a decision has to be made either to not let these rules be part of the relaxation or to classify the elements in some manner that shows that they represent a more unusual construction. An example of the classification is given in table 5.

NP-rules	NP left elements	NP core elements	NP right elements
NP -> Det <b>NN1</b>	Det	NN1	PP
NP -> Det Adj <b>NN2</b>	Adj	NN2	VP
NP -> Adj <b>NNP</b> PP		NNP	
NP -> <b>NN1</b> PP VP			

*Table 5. An example of the classification of left-, right and core elements of NP-rules.*

When the left and right (and middle) elements have been classified, they are made part of the robust parsing process. As I described in section 5.3, the rules in the initial grammar are relaxed by allowing fillers to appear in them. For controlling this relaxation, the refining constraint is to allow specific types of fillers, namely those that are classified as elements in the given category and position. For example a filler in a NP on the left side of the core is allowed if the filler is an element that is classified as a NP-left-side-element. An example is given in figure 14.

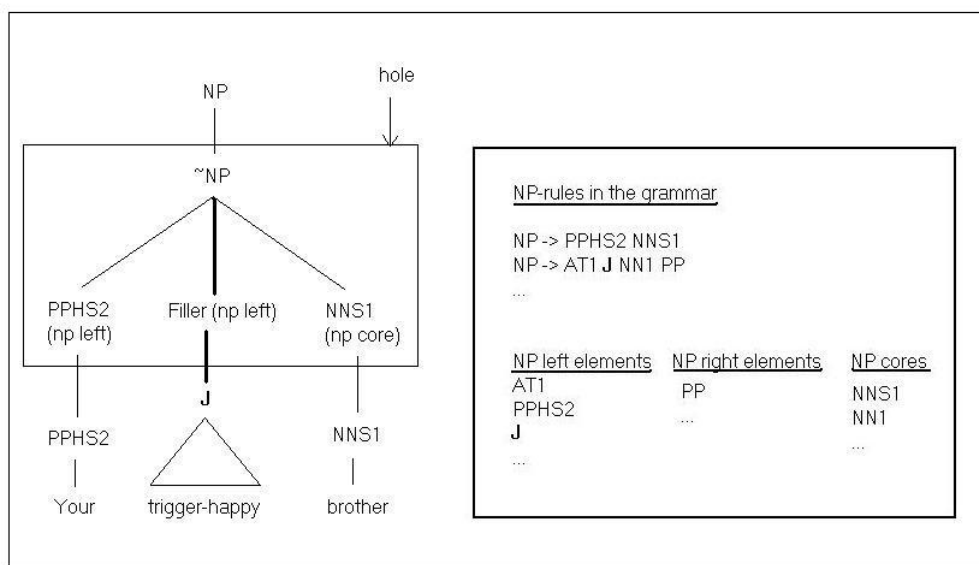


Fig 15. NP with a “filler” on the left side of the core. The filler is of category J, which has been classified as a NP left elements.

This method of dealing with extra-grammaticality may be criticized for being too restrictive. After all, it covers only one type of extra-grammaticality, which is the case when an extra element “fills out” the existing grammar rule. And those fillers have to be of specific types already occurring in the grammar. There are other forms of extra-grammaticality, for example the opposite situation when a phrase is too *short* to match any of the rules in the grammar. The *Minimum Distance Parsing approach* [9,10], which is based similar ideas, is more flexible by allowing three types of rule relaxation: insertions, deletions and substitutions. There is also no restriction on which rules can be relaxed. Although the techniques work well for small applications, they have been criticized for being too inefficient when applied on large grammars because the search space that the parser needs to explore is huge [17]. One of the purposes of my experiments is to investigate exactly how restrictive the method is. In section 7.2 I will discuss the results and provide some possible extensions that can be made to improve the coverage of the robust grammar.

In the next chapter I will say a few words about the assignment of probabilities to the rules that are specially produced for the robust grammar.

### ***5.5 Assigning probabilities to holes***

The advantage of using a probabilistic grammar in any context is that it provides a means of controlling the ranking of ambiguous solutions. The most effective way of assigning probabilities to grammar rules is by training from large annotated corpora. In the robust grammar described in this section, some “robust” rules are added to the initial grammar, to allow holes and fillers. The probabilities of those rules cannot be assigned by “training” because they don’t come from a corpus. Either they can be assigned manually or with some method that does it automatically.

The important thing is that the probabilities of the new rules should be assigned in such a way that the following conditions are filled in the parsing process:

- To prefer analyses with few holes
- To prefer holes with few fillers

To achieve this, the scores of those rules have to be strictly lower than the rules in the initial grammar. For example the rule NP → ~NP introducing a hole has to have a probability strictly lower than all other NP-rules.

In the next section I will describe the implementation of the robust grammar and how the probabilities were assigned.

## 6 Validation

In this section I present the methodology that is used for testing the techniques described in chapter 4 and 5. Chapter 6.1 describes the implementation of technique 2. The other technique has recently been implemented in a parallel project [20] and is available as a tool in the SLPtoolkit [21]. This tool and other tools that are used in the experiment are presented in Chapter 6.2. The data for the experiments are taken from two corpora, ATIS and Susanne.

Chapter 6.4 describes the methodology chosen for the tests. It's based on a comparison of the two techniques to reveal their advantages and disadvantages. In 6.5 I give some comments about the limitations of the chosen methodology.

### 6.1 Implementation

The “hole” strategy to robustness could be implemented in different ways, but the implementation I chose is purely grammatical. The grammar in this case is a probabilistic CFG, which is compatible with the SLPtoolkit [21].

The robust grammars are based on the grammars that are extracted from ATIS and Susanne. According to the observation I made that some grammatical categories are more likely to be the source of extra-grammaticality than others, I decided to limit the robustness to rules of the categories S, NP and VP and some closely related categories. For example, the S-rules in the Susanne corpus have subcategorization when there is a conjunction of two sentences. The subcategory is named S+ and the only difference between this category and S is that it starts with a conjunctive word such as “and” or “or” (see figure 15). This results in some very simple rules for the roots of the holes:

S -> ~S	S+ -> CC ~S
NP -> ~NP	VP -> ~VP

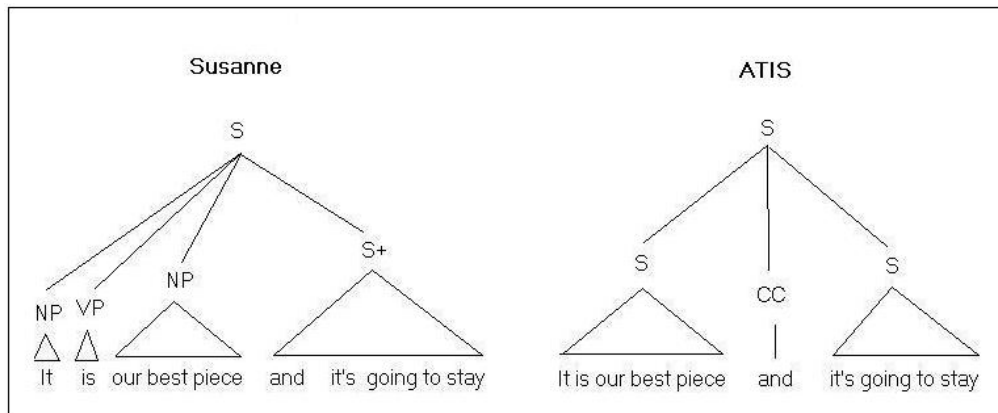


Fig 16. How conjunctions are represented in Susanne and in ATIS

As a second step, all the elements that occur in the rules of the chosen categories are classified as left-, right or core elements. This classification means the production of additional rules. One new rule is produced for each new type of element that occurs in a specific category and position. Table 6 shows an example of the procedure.

Grammar rules in the initial grammar	Rules that classify left- right and core elements
VP -> VVD	vp_core_VVD -> VVD
VP -> DA2 R VVN	vp_left_DA2 -> DA2
	...
NP -> ICS NN1	np_core_NN1 -> NN1
NP -> NN1 NN2	np_core_NN2 -> NN2
NP -> NN1 PP	...

Table 6. Example of the classification of left-, right- and core elements of a grammar.

To the left is the initial grammar, to the right are the new rules that are produced.

The next step is to select all the S, NP and VP-rules from the initial grammar and to write new, relaxed versions of them where the elements are marked by their positions and where fillers are allowed to occur between the elements. The smoothest way of writing such rules is to extend the initial rule so that fillers are

optional at all positions, the type of filler depending on where it occurs, as in the example in figure 17.

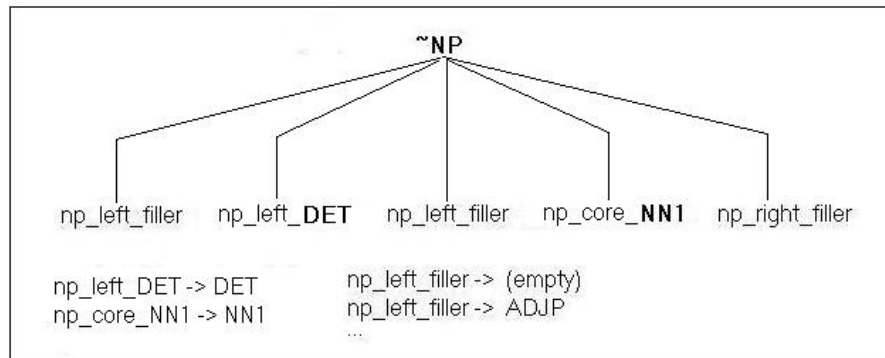


Fig 17. Example of relaxing a rule and allowing optional fillers at any position

The CFGs that are parsed in the SLPtoolkit environment are not allowed to have optional parts in the sense that rules can have empty right hand sides. An alternative, which is not very appealing, is to write every possible variation of the rule in which one or several fillers are present. That produces too many new rules. Instead one can go one level deeper in the structure and allow each element in the rule to consist of the original element followed by one filler (or several). Figure 18 shows how this is done.

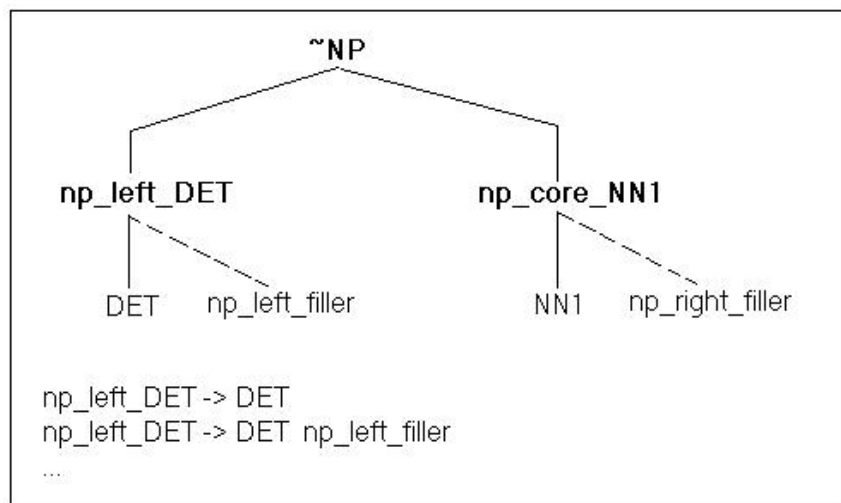


Fig 18. Alternative way of relaxing grammar rules. For each element classification rule one additional rule allows fillers.

One additional rule is needed to cover the cases where the filler appears at the left-most side of the phrase:

$\sim\text{NP} \rightarrow \text{np\_left\_filler } \sim\text{NP}$

What still remains now, before the robust grammar is complete, is to specify the elements that are allowed to be fillers in each category and position. In Chapter 5.4.3, I said that a filler is accepted if it has been classified as an element of the specific category and position where it occurs. For VPs and NPs, the fillers have been classified accordingly. For example:

$\text{np\_left\_filler} \rightarrow \text{DET}$       (from  $\text{np\_left\_DET} \rightarrow \text{DET}$ )  
 $\text{np\_right\_filler} \rightarrow \text{PP}$       (from  $\text{np\_right\_PP} \rightarrow \text{PP}$ )

For the S-category I have made an exception. Observations of the initial grammars show that the elements that occur to the left-, right- and middle of the core are very similar. This makes sense when considering that sentences are very flexible phrases where word order can be shifted to stress some particular words or phrases. Therefore *all* elements appearing in S-rules that are not part of the core (NP-VP) are specified as right-, left- and middle fillers.

Rules that don't contain a typical core have been ignored in this experiment. They are still present as regular CF-rules, but are not represented among the relaxed ones. The reason is that they are believed to cause overgeneration of erroneous trees rather than producing correct ones.

Assigning probabilities to all the new rules is not a simple task when you do it manually. The relaxed S, NP and VP-rules have inherited the probabilities of their correspondences in the initial grammar. This does not interfere with the probabilities of the initial rules, because the new rules are not actually of the

same categories as the initial ones but of new categories that mark that these are “almost recognized” structures (~S, ~NP and ~VP).

The rules that connect a hole to its mother node ( S -> ~S etc.) are given strictly lower probabilities than the rest of the rules. This will make solutions with fewer holes ranked higher than those with many holes. Solutions with no holes will always have better scores than solutions with holes, but since the robust grammar is only used for sentences that fail to be described by the initial grammar, there will be at least one hole in the full analysis.

The classification rules for left- and right elements as well as fillers have the same probability independently of their contents. Elements without a filler have a probability which is almost 1 and the ones with a filler are closer to 0.

np\_left\_**DET** -> **DET** (0.9999)

np\_left\_**DET** -> **DET** np\_left\_filler (0.0001)

Consequently holes with fewer occurrences of fillers are ranked higher than holes with many. To keep down the number of (unwanted) analyses, two or more fillers are not allowed to occur in a sequence, but the filler can be surrounded by commas, citation marks etc. For example:

s\_middle\_filler -> YC s\_middle\_filler YC (YC: comma)

This restriction can be relaxed in case no full analysis is derived, but it will also result in a larger search space.

Now I have described how the robust grammar is produced. In the next chapter I will present the syntactic tools that I have used during the project.



## 6.2 Tools

In this chapter I describe the tools that are used in the experiments with the two robust parsing techniques previously described. The tools are integrated in the SLPtoolkit, which is an NLP software library, developed at EPFL for internal use [21]. Most of the tools were implemented before the project was started, but several were developed during the project to make it possible to do experiments that are especially relevant in the context of robust parsing.

### 6.2.1 Initial tools

The tools that were previously implemented in the SLPtoolkit and have been used in this project are listed in table 7.

Name of tool (or option)	Use
Anagram	Stochastic bottom-up chart parser
-best	Return an analysis with the best score
-P	Analyses with a top node as root
-stat	Statistics on the parsed sentences
extractCFGfromcorpus	Extract grammar and lexicon from corpus
divisetreebank	Divide treebank in two parts
-prop	Select the proportion of each part
generephrase	Extract sentences from treebank
listlexique	Transform lexicon from binary to txt-format
listcompiledgram	Transform grammar from binary to txt-format
compilgram	Transform grammar from txt- to binary format
voir_arbres	Display analysis in “visual format”
selectarbre	Display two analyses in “visual format”
produce-results	Performance (accuracy, recall, coverage)

*Table 7. Tools and parsing options in the SLPtoolkit*

“Anagram” is the parsing tool, more precisely a stochastic bottom-up chart parser. During analysis it stores partial analyses in a specially designed data structure, from which full parse trees can be extracted efficiently.

Context free grammars and lexicons can either be written by hand or extracted automatically from tree banks with the “extractCFGfromcorpus” tool. The probabilities of the grammar rules are assigned during the extraction.

The “divisetreebank” command divides a tree bank in two parts, which is useful when experimenting with learning and test sets. The proportions of the two parts can be chosen specifically or the tree bank can be divided arbitrarily.

The sentences that are represented in bracketed trees in the tree bank can be extracted as strings of words with “generephrase”.

The anagram tool has several options for which information is to be extracted. The “best” option returns an analysis that has the probability of the most probable parse. There can be several solutions with the same probability and only one of them is displayed. Another option is to derive all the full parse trees, but for some sentences there can be several million solutions, so this option is not appropriate for all sentences. One choice, which is compatible with both of the other two, is to display only those trees for which the root is a top node.

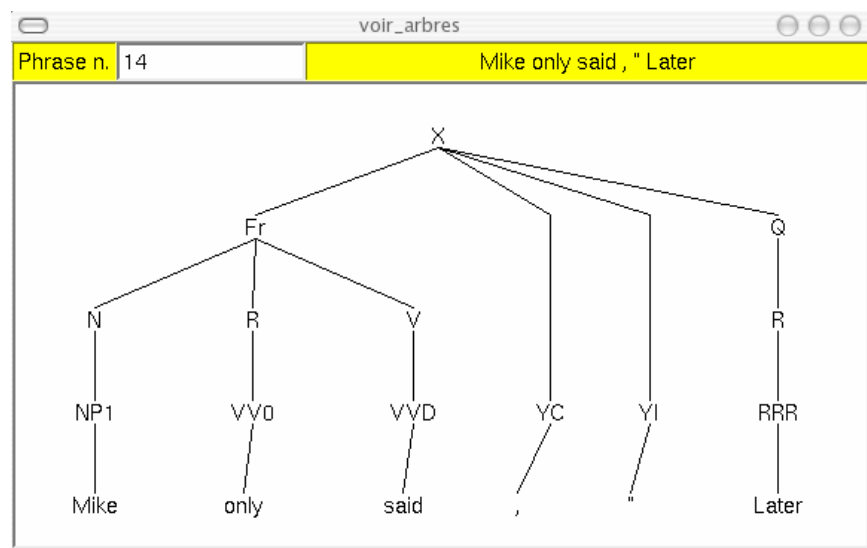
The “stat” option gives statistical information about the parsed sentence(s), e.g. the number of analyses.

When parsing a set of sentences with a grammar, it is possible to measure the performance by using the “produce-results” facility, which compares the most probable parse with the reference tree in the tree bank and gives percentages of accuracy, recall and coverage. This facility is unfortunately not helpful in the context of robust parsing because the trees that are produced with the two robust techniques have structures that are not present in the reference trees. They are the correct “robust” derivation trees and are therefore not identical to their corresponding reference trees.

Two tools exist for displaying trees in a visual form (the parser returns trees only in bracketed form). The first tool is “voir\_arbres”, which displays the trees in the regular fashion. With “selectarbre” two trees can be viewed at the same time, for example the reference tree and a “best” parse. The differences between the two analyses are highlighted with different colors.

### 6.2.2 Tools developed for the experiments with robust techniques

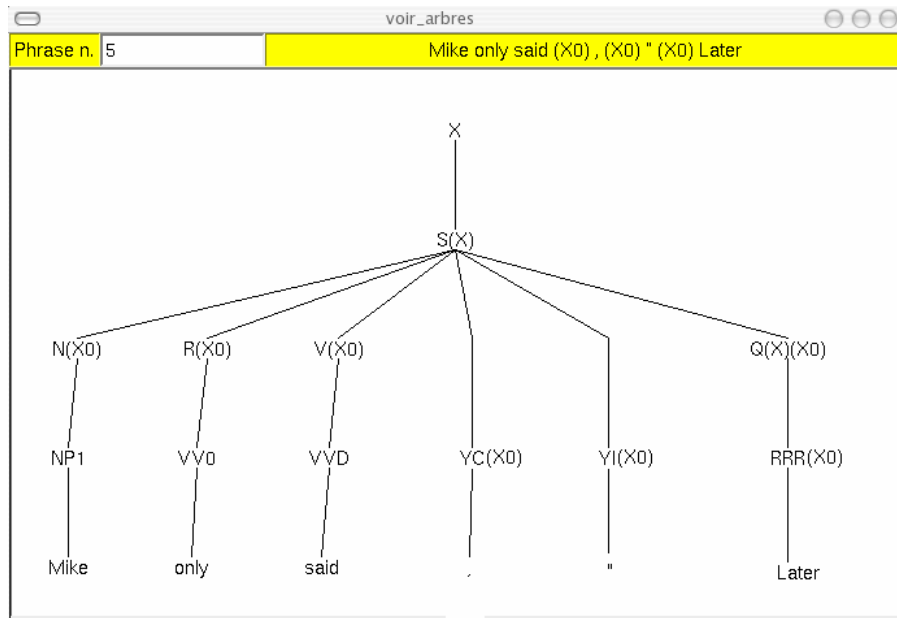
Four new tools have been especially implemented for experimenting with robust parsing. “Coverage” is a parsing option that returns the most probable optimal coverage of maximum trees from a forest of partial analyses. If there is a complete analysis, it will return the full tree that has the highest probability (or one of them), otherwise it will return an artificially produced full tree, where the partial analyses are glued together by a gluing rule, as described in section 4.2. An example of the output of this tool is given in figure 19.



*Fig 19. The output of the parsing option “coverage” viewed with the tool “voir\_arbres”*

A second tool, “tree\_with\_holes”, is a help for analyzing sentences for which one or several rules are missing. Given the reference tree of that sentence and a grammar, it displays the reference tree with holes. What it means is that rules

which are not in the grammar, but which are necessary to derive the reference tree are marked in the structure. An example of the output of “tree\_with\_holes” is given in figure 20.



*Fig 20. The output of the tool “tree\_with\_holes”. The rules that are missing in the grammar for producing the tree are marked with X for the left hand side and X0 for the right hand side of the rule.*

In order to know if a good parse is among the most probable parses, the parsing option “mpp” (Most Probable Parses) returns all the analyzes that share the score of the most probable parse.

Finally, there is a tool which tests if a specific tree is derived by the grammar. Further it tests if the tree is scored as (one of) the most probable parse(s) for that sentence. This tool has not been tested yet.

### **6.3 Corpora**

The data, which is used for testing the robust parsing techniques, is taken from two corpora, both treebanks. The ATIS (Air Travel Information System) corpus is

a cleaned version of the original ATIS corpus, which means that language disfluencies and other speech errors are removed. The utterances were originally recorded from people who were interacting with an information system with the purpose of booking flight tickets. The variation of expressions is therefore smaller than can be expected from other corpora, where the sentences are collected from wider domains.

Susanne (Surface and Underlying Structural Analysis of Natural English) is a corpus with written American English data. The sentences are a subset of the million-word Brown Corpus [22]. Some characteristics of the ATIS and Susanne treebanks are given in table 8 [7].

	<b>ATIS</b>	<b>Susanne</b>
Number of sentences	1,381	3,981
Number of CF rules	1,029	8,810
Number of non-terminals	40	469
Number of terminals	1,167	10,247
Number of POS tags	38	122
Average sentence length	12.5	12.9
Average number of CF rules per sentence	23.3	23.8
Max depth	17	17

*Table 8. Some characteristics of the two corpora used for the experiments.*

The characteristics of the two corpora are very different. Susanne has a much richer annotation of syntactic structures than ATIS, which leads to a significantly larger grammar and more ambiguous solutions in the parsing process. These characteristics are taken into consideration in the experiments.

## **6.4 Methodology**

In this chapter I present the methodology that has been applied in the experiments. I describe how the data was prepared for the tests, how each technique is tested and evaluated separately and how they are compared. In the

following chapter I also give some comments about the limitations with this methodology.

#### 6.4.1 Generating grammar and lexicon

The experiments that are performed with the two robust techniques are performed on the two corpora previously described. With the tools in the SLPtoolkit, each treebank is divided into a learning set and a test set. The lexicon is extracted from the whole treebank in both cases, because the intention is not to deal with out-of-vocabulary words. The learning set of each corpus is used for extracting a grammar. It has to be done in such a way that a reasonable part of the test set is not covered by the grammar and needs to be treated in a robust way. In order to be able to parse the test set, the sentences have to be extracted in string format from the tree bank by using the “generephrase” utility.

The differences in the characteristics of ATIS and Susanne make it necessary to divide the treebanks in different ways. The details of the division are given in table 9.

	Learning set (%)	Test set (%)	Uncovered sentences
ATIS	10%	90%	89 (7%)
Susanne	50%	50%	250 (13%)

*Table 9. The proportions of learning and test set for the ATIS and Susanne corpus. The third column shows the amount of sentences in the test set that are not described by the grammar extracted from the learning set.*

ATIS cannot have a larger learning set because that would make the undergeneration non-existent. In the Susanne corpus, on the other hand, 77% of the rules appear only once. Therefore a smaller learning set is less suitable.

### **6.4.2 Testing robust technique 1: most probable optimal coverage**

The first experiment I do is to parse the test sentences with the –coverage option which gives the “most probable optimal coverage”. This option of the parsing tool is described in 6.2.2. Only the test sentences that are uncovered by the extracted grammars are tested. The resulting artificial trees are evaluated by dividing them into three groups.

1. Correct partial analyses. For all the parts of the input that have some analysis, the analysis has to match a subtree in the reference tree.
2. Not correct partial analyses but still a useful close analysis of the input
3. Not acceptable analysis of the input. The sentence is falsely interpreted and the true meaning is therefore not preserved

Examples of each type of tree are given in figure 21. I should mention that the decision of whether an analysis can be classified as useful or not is not completely obvious when doing the analysis by hand. It makes more sense to measure the usefulness by looking at the system as whole. For example, if the parser is a part of an automatic translation application, the usefulness can be measured by the quality of the translation, or if it is a dialogue system, the success rate can be judged by the responses that the system gives to the user input. One has to take into account that the other modules in the system affect the behavior as well, so when looking at the final output, it is not the structural analyzer alone that can be the cause for a bad result.

In this case, the “usefulness” of an analysis is determined by lexical correctness and appropriate segmentation of the input into phrases. In uncertain cases, linguistic intuition is applied as the deciding factor.

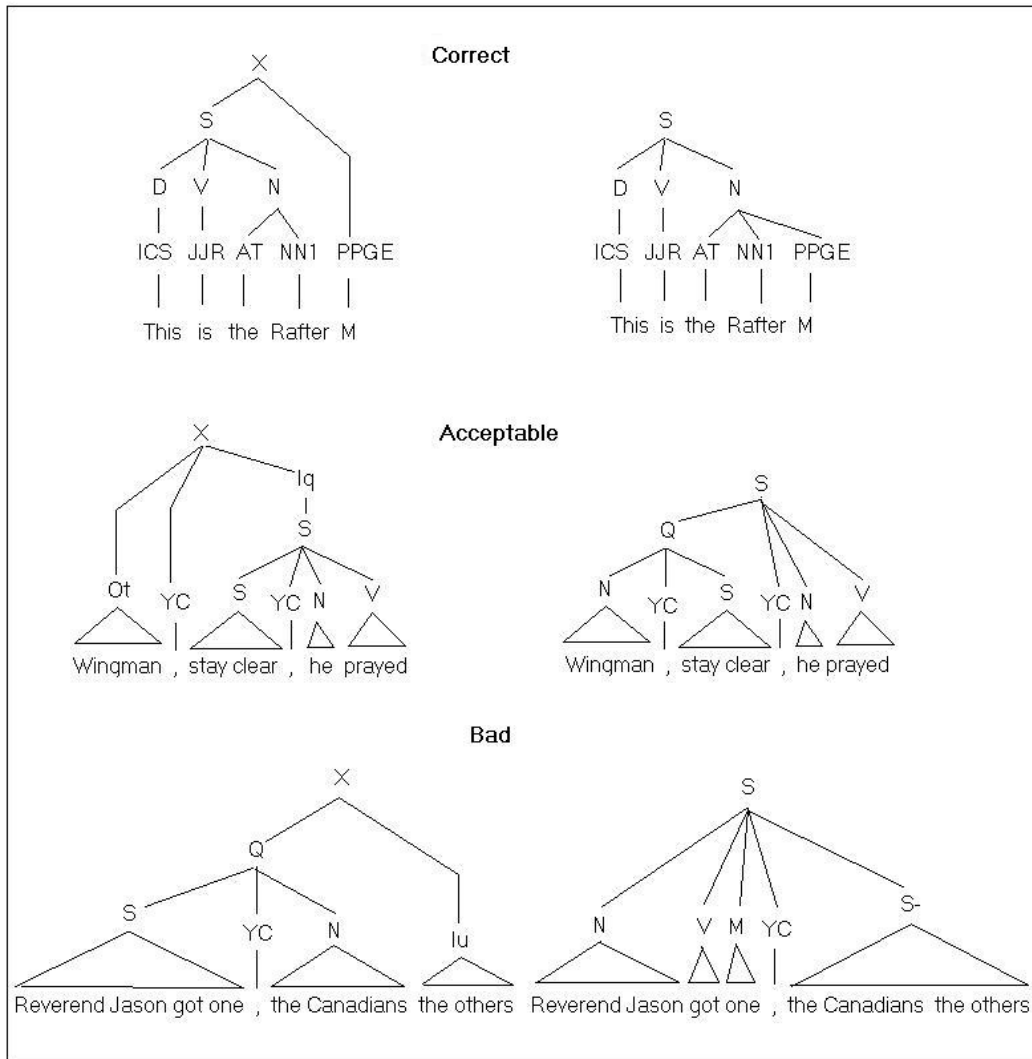


Fig 21. Examples of a correct optimal coverage, an acceptable one and a bad one. For each example, the reference tree is given to the right.

### 6.4.3 Testing robust technique 2: most probable full tree with holes

The second experiment demands some work. First of all, the robust grammars need to be created, as described in 6.1. They are based on the grammars that are extracted from the learning sets of the two corpora, and they may be produced automatically in the future, but for this relatively small experiment, I have done it manually.



The tests with the robust grammar have two main purposes. The first is to investigate how well the test sentences actually match the restricted model of robustness, which I described in 5.4. Does the assumption hold that holes should only be allowed in S, NP and VP categories? The second purpose is to find out if it leads to a high accuracy if holes are restricted to be “almost recognized” structures where specific types of fillers appear at specific positions. Maybe the restriction is too strong and will lead to a bad coverage instead.

To answer these questions, the test sentences are observed with the “tree\_with\_holes” tool which shows where the holes should appear and which missing grammar rules they should compensate. When the sentences are parsed with the robust grammar, the number of sentences that fail are counted and observed closely to see the cause of the failure and what might improve the coverage of the grammar. For the other sentences, an evaluation is made like the one for the first technique. The analyses are divided into three groups.

1. Correct analysis. The holes compensate missing rules and the rest of the tree is correct.
2. Useful close analysis. The hole or some other part of the tree is not correctly parsed, but the meaning of the sentence is largely preserved
3. Not acceptable analysis. The interpretation is so far from the correct one that the original meaning of the sentence is not preserved

Examples of each case is given in figure 22.

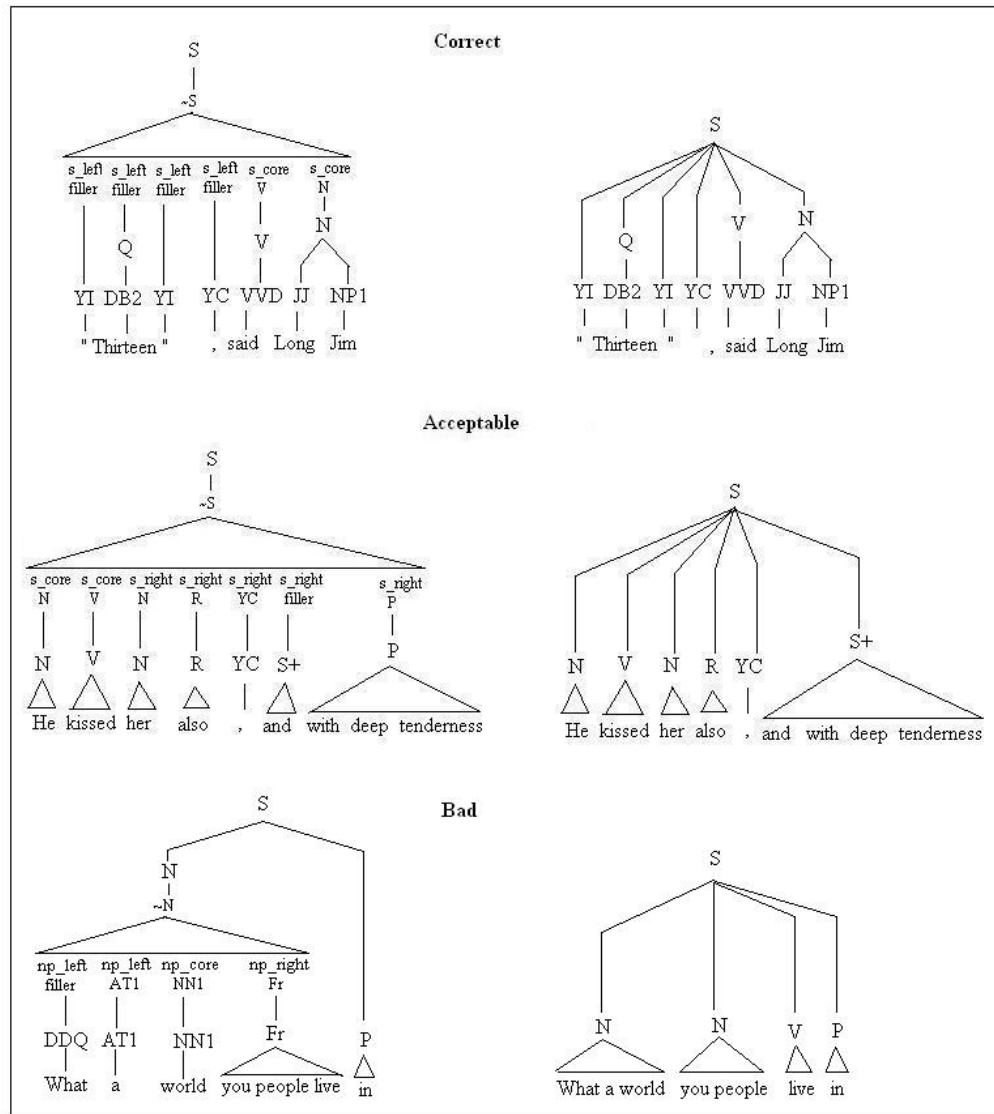


Fig 22. Example of a correctly constructed tree with hole, an acceptable one and a bad one. For each example, the reference tree is given to the right.

#### 6.4.4 Comparison and analysis of the two techniques

When the test sentences have been parsed with both robust techniques and on both corpora, a comparison is made too see if there are any differences between the two approaches. The comparison involves the following questions:

- How many correct or useful analyses versus bad analyses are there (for each corpus)?

- How many of the sentences are well parsed by both techniques?
- How many are badly parsed by both?
- How good is the analysis returned by technique 1 when a sentence is not covered by technique 2?

The results are then analyzed. The purpose is to explore the differences in behavior, if there is any, and to understand what caused it. More specifically, the questions that are in focus are:

- Can some indicators be found to explain why some sentences succeed with technique 1 but not with technique 2?
- What indicators can be found in the opposite situation, when technique 2 succeeds and technique 1 does not?
- What are the reasons that both techniques perform badly for some sentences?
- If there is a difference between the techniques concerning the performance on the different corpora, what is the cause?

The main factors that are studied when analyzing the results are:

- The number of rules that are missing in the initial grammar for producing the correct analysis for a specific sentence
- The category of missing rules
- The level of ambiguity for the part of the sentence that is the source of the extra- grammaticality
- The type of top node (if it is of category S or something else)

If some other factors can be found to explain differences, these are reported too.

## **6.5 Some comments about the methodology**

Before presenting the experimental results and the analysis of them, I should say a few words about how the methodology was chosen. It is perhaps not the best one and certainly not the only one possible. I have chosen to use the SLPtoolkit, which means that I've also chosen to take a specific direction both concerning the theory and the way of testing it. The choice of corpora has additionally had an influence.

The approach based on gluing partial analyses was developed independently of any corpus, but the data in ATIS and Susanne was studied both before and after the development of the other technique. Studying two different types of corpora is good, because the ideas that come out have bigger chances of being applicable in various domains. But ATIS and Susanne both have rather extreme characteristics. Susanne has a very detailed and rich annotation, which I mentioned earlier. This is not necessarily a good thing when experimenting with robust techniques. The ambiguity complicates the efforts of producing useful analyses. ATIS on the other hand is so homogeneous that it is difficult to find interesting examples of extra-grammaticality.

The tools provided by the SLPtoolkit have to some extent influenced the whole way of looking at the problem of extra-grammaticality. The solution is already given, in the form of a reference tree in the tree bank, and the task is to arrive at that solution. But many other methods are possible. For example, corpora that contain pure text with no syntactic annotation have to be used in other ways.

Also the grammar does not have to be automatically generated from a tree bank and it doesn't have to be probabilistic. In fact, there are many grammar formalisms that are more sophisticated than CFGs, but by using SLPtoolkit one is limited to this type of grammar.

An additional thing that I should mention is that the “Most Probable Parses”-tool was implemented after the preliminary tests for the second approach with the “hole”-concept. This may have had some impact on the impressions that were gathered during that testing phase. The “best”-option that was used instead, displays only one of potentially many “most probable parses” and does that arbitrarily.

I have mentioned some aspects that have influenced the choice of methodology previously described. Now I will present the results of my experiments and discuss the difference in behavior between the two techniques.

## 7 Experimental Results

In this section I present the experimental results from the tests described in the last chapter. In 7.2 I discuss the results and in 7.3 I propose some additional experiments that could not be done because of the limited time space.

### 7.1 Results

The experimental results from the tests are given in table 9. Only the performance of the second, robust, phase of the parsing process is measured, which means that only the sentences that failed with the initial grammar are tested. For the Susanne corpus, a subset (56%) of the 250 sentences was selected for the tests because of the amount of work that it took to analyze the results by hand.

	ATIS				Susanne			
	Correct	Useful	Bad	No analysis	Correct	Useful	Bad	No analysis
Technique 1: coverage	10%	60%	30%	-	16%	29%	55%	-
Technique 2: holes	24%	36%	9%	31%	40%	17%	33%	10%

*Table 10. Experimental results. Percentage of correct, useful and bad analyses with the two robust techniques, tested on ATIS and Susanne.*

A comparison of the two techniques is summarized in table 10. These numbers indicate how often the techniques were successful (or unsuccessful) for the same sentences and how often a sentence was analyzed better with one technique than with another.

	Technique 1 better (%)	Technique 2 better (%)	Both successful (%)	Both unsuccessful (%)
ATIS	29	33	24	14
Susanne	17	30	27	26

*Table 11. Comparison of the two techniques. Percentage of sentences that had a better analysis with one technique than with the other and percentage of sentences that had an equally successful analysis with both techniques*

## **7.2 Analysis**

The results from the experiments with the two robust parsing techniques show that the second one based on “holes” produces the correct robust analysis more often than the first one based on the optimal coverage. In the comparison of the two corpora, there is a difference. When testing on ATIS, technique 1 is able to provide a useful close analysis in most cases, while technique 2 suffers badly from undergeneration. Tests with Susanne give the opposite results: Technique 1 is not able to produce many useful analyses and Technique 2 has not so much undergeneration. Both produce a large number of bad analyses though. For approach 1 the bad analyses are more than 50%. For approach 2 the number is somewhat lower than 50% when including both bad analyses and cases when no analysis was returned.

There was an indicative pattern related to technique 2. For about half of the sentences that had a bad or no analysis, the missing rule(s) were of other categories than S, NP or VP. In ATIS, those rules were often of category PP or ADJP, but in Susanne there was no particular category that occurred significantly more often than some other. This can be explained by the grammatical annotation in Susanne. The number of non-terminals appearing in the corpus is 469, which is quite high compared to ATIS with only 40. There are simply very many possibilities as to which type of rule is missing in the grammar.

The relatively low coverage of technique 2 when testing on ATIS can be explained by the homogeneity of the corpus. It took a division of 10% learning set and 90% test set to even get an undergeneration as high as 7%. The variation of syntactic constructions is small and therefore the lists of possible left and right fillers for the NP, VP and S categories are very short.

One pattern that was connected to both techniques was that the majority of sentences that were badly parsed lacked three or more rules in the initial grammar for deriving the reference tree.

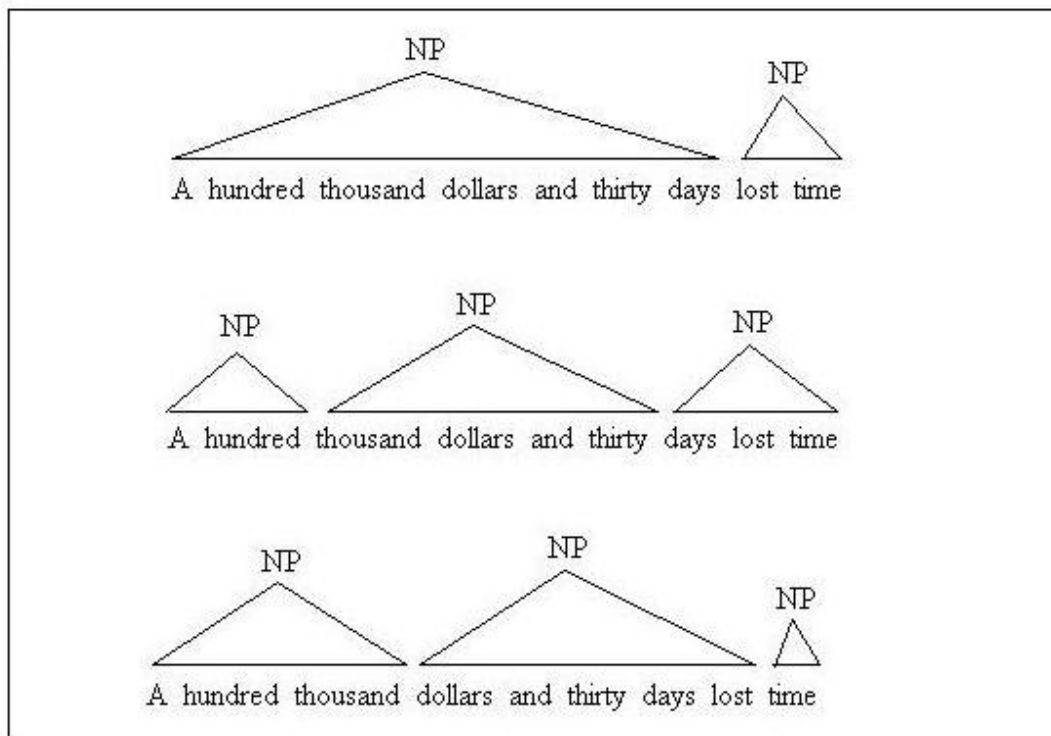
In some of the cases when technique 1 behaved better than technique 2, the root of the reference tree was different from S. The parsing options that were chosen when parsing with the robust grammar in technique 2 were to display all the most probable parses but only those for which the root is the top category (S). Because of this parsing option, some analyses that had a root other than S could never be among the most probable parses, even if they had the best score. Instead there would be analyses with S as a root if the robust grammar were able to derive them. Some of the sentences that had no analysis at all may have failed because of this parsing option. It is questionable though if the results would be better without it. While giving a possibly larger coverage, the accuracy may be affected. After all, most of the sentences in the tests (at least for Susanne) were correctly parsed, and many of those that were not, were bad because of other reasons.

The most visible cause for bad analyses with the Susanne corpus is the ambiguity that lies in the sentences as well as in the extracted grammar and lexicon. This problem is unsolved in both robust techniques. Some of the bad analyses are more caused by the sentence than the grammar. Consider for example a sentence like:

“It cost us a hundred thousand dollars and thirty days lost time to fix them”.

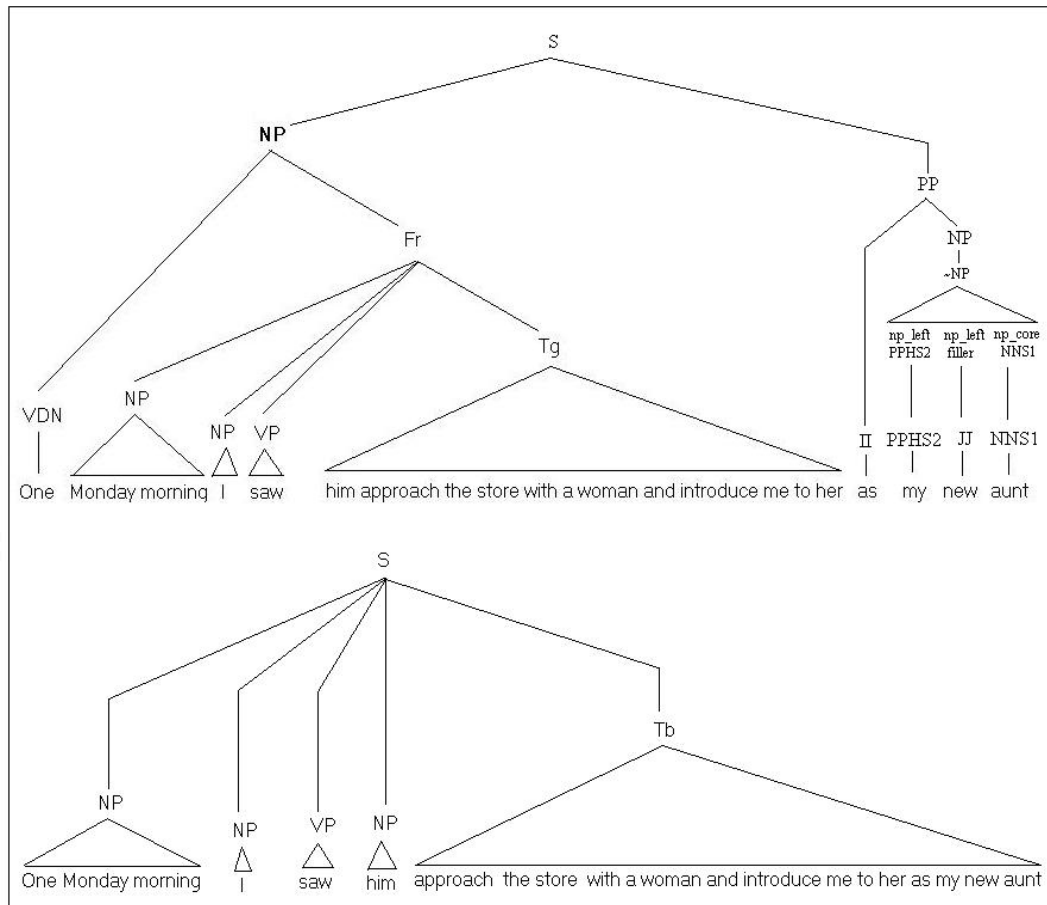
The nominal phrase “a hundred thousand dollars and thirty days lost time” can be parsed in many different ways (see figure 23), and for knowing which analysis is the correct one, some particular methods for disambiguation are needed.





*Fig 23. Three analyses of the ambiguous NP "A hundred thousand dollars and thirty days lost time". These are not the only possible ones.*

But there are examples where it is the grammar rather than the sentence that causes the bad "best" analysis. In the Susanne grammar a typical phenomenon is that large parts of the sentence that should be VPs and S-complements are instead parts of a NP. The grammar describes numerous kinds of constructions that are valid complements in NPs. There are even some rules where complements can exist without the core. Figure 24 demonstrates a sentence parsed with technique 2 that has a very bad "best" analysis.



*Fig 24. A bad analysis caused by the ambiguity in the grammar extracted from Susanne. On top is the analysis returned by the robust technique based on holes. Below is the reference tree.*

In this example one can see, that although the hole was correctly produced, the analysis in whole is not satisfactory. The same problem exists for both techniques. The conclusion that can be drawn from this is that some specific method for disambiguation is important, independently of how the robust component is implemented in the parser. The same conclusion was made by Carroll & Briscoe, which I mentioned in section 2.3. Robust parsing should not only provide a better coverage for the input but also be able to handle ambiguity in an effective manner.

In short, the robust techniques that were tested in this project have the following properties:

Technique1:

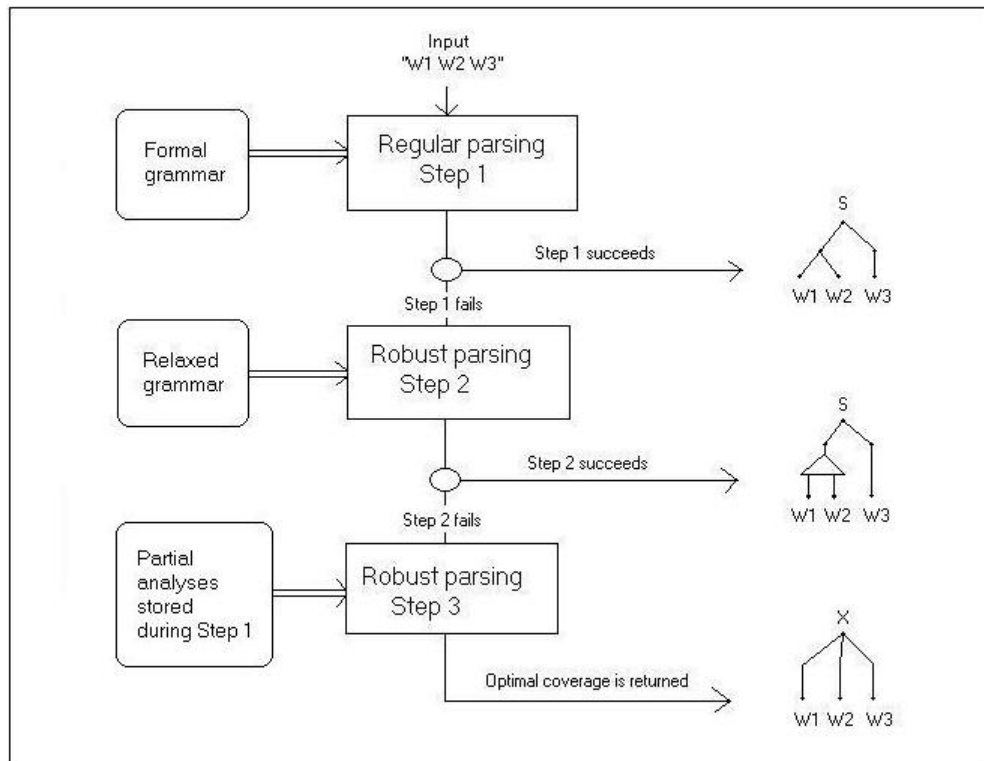
- All sentences have some analysis
- Sentences that are almost covered by the grammar are not as likely to be parsed correctly as with technique 2, but a useful close analysis is returned in many of the cases
- Sentences for which three or more rules are missing are more likely to have a useful close analysis than with technique 2 but also to have an analysis with many lexical trees.
- For sentences that have no analysis with technique 2, an acceptable analysis is returned in more than half of the cases

Technique 2:

- Sentences that are almost covered by the grammar (in Susanne the majority!) are very likely to be correctly parsed
- Sentences for which three or more rules are missing in the initial grammar are more likely to have incorrect analyses or no analysis at all
- Sentences for which the missing rules are other than S, NP or VP are also more likely to have bad analyses or no analysis at all
- A grammar that covers a rich variety of syntactic structures makes the robust grammar more flexible and gives a better coverage

Both techniques suffer from the ambiguity problem.

In summary, one can say that technique 1 guarantees full coverage but technique 2 has a better accuracy. A solution that would take the advantages of both approaches is to apply them in a sequence, one after the other. The architecture that the structural analyzer would have is shown in figure 25.



*Fig 25. The architecture of a parser that analyses in three steps (step 2 and 3 only if the previous step failed).*

### 7.3 Further experiments

In this chapter I describe some experiments that were not done with technique 2, because the time space was limited, but which could improve the performance of the technique.

1. Test other forms of rule relaxation in the robust grammar,

The basic idea with the rule relaxation is to choose a rule from the initial grammar that best matches the input and to allow some minor differences, which in this implementation is the insertion of fillers. Other possibilities would be to remove one or two elements from the initial rule (deletion), or to replace one element by another (substitution). The second option is particularly interesting for the cores. For example a VP-rule may match the input almost perfectly except that the core is a different kind of verb (VBZ instead of a VB). For avoiding too many unwanted

analyses, the deletion and substitution should be done in the same restrictive way as for the insertions, by allowing the modifications only with elements that are classified in the specific category and position.

## 2. Add more categories in the robust grammar

The robust grammar was implemented only for the categories S, NP, and VP. When analyzing the test results many examples were found of other categories of missing rules. The question that arises is if the coverage is improved by such an addition and if that improvement means a cost of accuracy and computational efficiency.

## 3. Elaborate with the probabilities of the rules

The rules in the robust grammar that correspond to the classification of fillers (NP left fillers, NP right fillers, S fillers etc) all have the same probabilities. But in real life some of them may be more probable than others. Since the probabilities at this stage have to be assigned manually, it is difficult to do any realistic estimation of what the values should be, but some guidelines can be drawn from the initial grammar:

- How often is the category of the filler represented in the grammar?
- Is it unique in one specific type of rules or does it occur a bit everywhere?
- Is it more often on the left or the right of the core?

If the probabilities are based on this type of information, the assignment of probabilities to the rules could be made automatically.

Now I have described the test results from my experiments, given some answers to the behavior of the two robust techniques, and made some suggestions for further experiments. In the next section I will summarize the whole work with robust parsing and give some prospects for future work.

## 8 Conclusions and Future Work

This report presents two approaches to robust stochastic parsing. One is based on the selection of the most probable optimal maximum coverage. The other derives full trees with holes that are produced with a rule relaxation strategy. The experimental results show that the first technique guarantees full coverage whereas the second is superior in terms of accuracy. Applying them both in a sequence has several advantages:

- The accuracy of the second technique is preserved and full coverage is accounted for
- No language specific, domain specific or semantic information is used in the parsing process, which makes the approach suitable for general-purpose
- Restrictions on the grammatical categories of holes and fillers make the rule relaxation strategy more efficient than other *Minimum Distance Parsing* techniques, where every possible relaxation is considered.

The techniques are nevertheless far from perfect, and further work is required to solve (among others) the following issues:

- The error-analysis implies that many of the incorrect analyses were due to relying purely on a probabilistic model for disambiguation
- There was no method for dealing with lexical ambiguity
- The restrictions on the flexibility of technique 2 were based on an observation of 40 examples of extra-grammatical sentences and proved to cause a varying degree of undergeneration depending on the initial grammar
- The approaches were only tested on two corpora, which means that the tests were limited to those phenomena occurring in the corpora

- Different forms of ungrammaticality, such as disfluencies, were not considered at all and they are an essential issue in spoken language processing.

The primary task, before moving further with these questions, is to implement a method for automatically generating a relaxed grammar from an initial grammar. It should be implemented in such a way that it is easy to tune the flexibility of the relaxed grammar for experimental purposes.

## 9 References

- [1] S Aït-Mokhtar and J-P Chanod (1997) Incremental Finite-State Parsing. Proceedings of ANLP'97, Washington.
- [2] J Bear, J Dowding & Shriberg (1992) Integrating multiple knowledge sources for the detection and correction of repairs in human--computer dialogue. Proceedings of 30th ACL.
- [3] R. Bod (1998). Beyond Grammar: An Experience-Based Theory of Language. CSLI Publications, Stanford.
- [4] J Carroll and E Briscoe (1996) Robust parsing - a brief overview. In J. Carroll (ed.), Proceedings of the Workshop on Robust Parsing at the 8th European Summer School in Logic, Language and Information (ESSLLI'96), Report CSRP 435, COGS, University of Sussex.
- [5] J-P Chanod (2000). Robust Parsing and Beyond. G.V. NOORD & J. JUNQUA, Eds., Robustness in Language Technology, Kluwer.
- [6] J.-C. Chappelier and M. Rajman. (1998) A generalized cyk algorithm for parsing stochastic CFG. In Proceedings of Tabulation in Parsing and Deduction (TAPD'98), Paris.
- [7] J.-. Chappelier and M. Rajman. (2000) Polynomial tree-substitution grammars: an efficient framework for Data-Oriented Parsing. In Proc. Of Recent Advances in Natural Language Processing (RANLP 2001), pp.65-71.



- [8] P Heeman and J Allen (1994) Detecting and correcting speech repairs. In Proceedings of the 32 Annual Meeting of the Association for Computational Linguistics, pages 295-- 302, Las Cruces, New Mexico, June 1994.
- [9] D R Hipp (1992) Design and development of spoken natural language dialog parsing systems. PhD thesis, Department of Computer Science, Duke University.
- [10] Lehman, J. F (1989) Adaptive Parsing: Self-Extending Natural Language Interfaces. Ph.D. thesis, School of Computer Science, Carnegie Mellon University. CMU-CS-89-191.
- [11] C. Manning and H. Schuetze (1999) Foundations of Statistical Natural Language Processing. Cambridge, The MIT Press.
- [12] M Marcus, B Santorini, and M.A. Marcinkiewicz (1994) Building a large annotated corpus of English: The Penn treebank. Computational Linguistics, 19(2):313--330, 1994.
- [13] D McKelvie (1998) The Syntax of Disfluency in Spontaneous Spoken Language. HCRC Technical Report, RP-95, 1998. University of Edinburgh.
- [14] G van Noord, G Bouma, R Koeling, and M-J Nederhof.(1999) Robust grammatical analysis for spoken dialogue systems. Journal of Natural Language Engineering.
- [15] E Oltmans. (2000) A Knowledge-Based Approach to Robust Parsing. PhD thesis, University of Twente, Enschede, The Netherlands.
- [16] S L Oviatt (1995). Predicting spoken disfluencies during human-- computer interaction. Computer Speech and Language, 9, pp.19--35.

- [17] C P Rose and A Lavie (2001) "Balancing Robustness and Efficiency in Unification-Augmented Context-Free Parsers for Large Practical Applications", *Robustness in Language and Speech Technology*, J. C. Junqua and G. Van Noord (eds.), *Veronis and Ide Series*, Kluwer Academic Press.
- [18] E Shriberg (1996). *Disfluencies in Switchboard*. Proceedings of 4th ICSLP.
- [19] K L Worm and C J Rupp (1998). *Towards robust understanding of speech by combination of partial analyses*. In *Proc. of the 13th ECAI*, Brighton, UK.
- [20] Vladimír Kadlec, Jean-Cédric Chappelier and Martin Rajman. *Tool for robust stochastic parsing using optimal maximum coverage*. Technical Report. Swiss Federal Institute of Technology (EPFL), Lausanne (Switzerland). To appear 2004.
- [21] The SLPtoolkit: <http://liawww.epfl.ch/~chaps/SlpTk/>
- [22] G Sampson. *The Susanne analytic scheme*:  
<http://www.grsampson.net/RSue.html>

## **Appendix**

A: Robust grammar ATIS

B: Robust grammar Susanne

C: The syntactic annotation for Susanne

## A: Robust grammar ATIS

The tables below show examples of rules that were added to the automatically extracted ATIS grammar in the experiment with “Robust technique 2”, described in section 5 and 6. The probabilities have been assigned by hand except for those rules that were copied from the initial grammar and renamed to enable relaxation, e.g. ~S (S) and s\_core\_NP (NP). They have inherited the probabilities from the original rules.

### A1. Rules for deriving holes

Description	Example
Rules introducing holes	S -> ~S (0.00000001) NP -> ~NP (0.00000001) VP -> ~VP (0.00000001)
Rules from the initial grammar with renamed LHS and RHS to enable relaxation	~S -> s_l_VB s_core_S (0.000155039) ~S -> s_core_NP s_core_VP (0.00124031) ~S -> s_core_NP s_core_VP s_r_VB (0.0000775194) ~S -> s_core_NP s_m_MD s_core_VP (0.00116279) ~S -> s_core_NP s_m_MD s_core_VP s_r_RB (0.0000775194) ~S -> s_core_NP s_m_MD s_m_RB s_core_VP (0.000155039) ~S -> s_core_S , s_r_UH (0.0000775194) ~NP -> np_l_PRP np_core_NNS np_r_PP (0.0000139665) ~NP -> np_core_NNP np_core_NNP (0.000377095) ~NP -> np_core_NNP np_core_NNP np_r_CD (0.0000558659) ~NP -> np_core_NNP (0.00167598) ~NP -> np_core_NNP np_r_ADJP (0.0000698324) ~NP -> np_l_DT np_core_NNS np_core_NN (0.0000139665) ~NP -> np_l_PRP\$ np_core_NN (0.000027933) ~VP -> vp_core_VB vp_r_NP (0.00205405) ~VP -> vp_core_VB vp_r_NP vp_r_PP (0.000162162) ~VP -> vp_core_VB vp_r_S (0.000648649) ~VP -> vp_core_VB (0.000108108) ~VP -> vp_core_VBZ vp_r_ADJP (0.0000540541) ~VP -> vp_core_VBG vp_r_PP (0.00027027) ~VP -> vp_core_VBP vp_r_SBAR (0.0000540541)
S left elements	s_l_VB -> VB (0.9999) s_l_RB -> RB (0.9999)
S left elements followed by left filler	s_l_VB -> VB lf_s (0.0001) s_l_RB -> RB lf_s (0.0001)
S core elements	s_core_NP -> NP (0.9999) s_core_VP -> VP (0.9999) s_core_S -> S (0.9999)
S core elements followed by (middle and right) fillers	s_core_NP -> NP mf_s (0.0001) s_core_VP -> VP rf_s (0.0001) s_core_S -> S rf_s (0.0001)

## A2. Rules for classifying fillers

Description	Example
Left filler S	lf_s -> PP (0.0000001) lf_s -> NP (0.0000001) lf_s -> VB (0.0000001) lf_s -> UH (0.0000001) lf_s -> INTJ (0.0000001) lf_s -> RB (0.0000001)
Right filler S	rf_s -> PP (0.0000001) rf_s -> NP (0.0000001) rf_s -> VB (0.0000001) rf_s -> UH (0.0000001) rf_s -> INTJ (0.0000001) rf_s -> RB (0.0000001)
Left filler NP	lf_np -> PRP (0.0000001) lf_np -> PRP\$ (0.0000001) lf_np -> ADJP (0.0000001) lf_np -> JJ (0.0000001) lf_np -> CD (0.0000001) lf_np -> DT (0.0000001) lf_np -> PDT (0.0000001)
Right filler NP	rf_np -> PP (0.0000001) rf_np -> ADJP (0.0000001) rf_np -> RB (0.0000001) rf_np -> NP (0.0000001) rf_np -> VP (0.0000001) rf_np -> SBAR (0.0000001)
Right filler VP	rf_vp -> NP (0.0000001) rf_vp -> PP (0.0000001) rf_vp -> ADJP (0.0000001) rf_vp -> ADVP (0.0000001) rf_vp -> RB (0.0000001) rf_vp -> S (0.0000001) rf_vp -> SBAR (0.0000001) rf_vp -> VP (0.0000001)

## B: Robust grammar Susanne

The tables in B1 and B2 below show examples of rules that were added to the extracted Susanne grammar in the experiment with “Robust technique 2” described in section 5 and 6. Like for the ATIS grammar, the probabilities have been assigned by hand except for those rules that were copied from the initial grammar and renamed to enable relaxation.

### B1. Rules for deriving holes

Description	Example
Rules that introduce holes	S -> ~S (0.0000001) S+ -> ~S+ (0.0000001) S- -> ~S- (0.0000001) N -> ~N (0.0000001) N+ -> ~N+ (0.0000001) V -> ~V (0.0000001)
Rules from the initial grammar, renamed to enable relaxation	~S -> sCore_N sCore_V sr_R (0.000013013) ~S -> sCore_N sCore_V sr_Ti (0.032032) ~S -> sl_R sCore_N sCore_V (0.000500501) ~S -> sCore_V sCore_N sr_J (0.001001)  ~N -> nCore_NN2 (0.0107367) ~N -> nCore_NN2 nr_P (0.0057813) ~N -> nl_DB2 nCore_NNU (0.0024777) ~N -> nl_AT1 nCore_NN1 nr_Ti (0.0008259)  ~V -> vCore_VBDZ (0.0441215) ~V -> vCore_VBDZ vr_CS (0.00337399) ~V -> vl_CCB vCore_VVN (0.00674799) ~V -> vCore_VBR vr_VVN (0.00493122)
NP left elements	nl_JJ -> JJ (0.999999) nl_CS -> CS (0.999999) nl_DB2 -> DB2 (0.999999) nl_AT1 -> AT1 (0.999999)
NP left elements followed by filler	nl_JJ -> JJ lfN (0.000001) nl_CS -> CS lfN (0.000001) nl_DB2 -> DB2 lfN (0.000001) nl_AT1 -> AT1 lfN (0.000001)
NP core elements	nCore_NNU -> NNU (0.999999) nCore_NNU2 -> NNU2 (0.999999) nCore_NN1 -> NN1 (0.999999) nCore_NNT1 -> NNT1 (0.999999) nCore_NNT2 -> NNT2 (0.999999)
NP core elements followed by filler	nCore_NNU -> NNU rfN (0.000001) nCore_NNU2 -> NNU2 rfN (0.000001) nCore_NN1 -> NN1 rfN (0.000001) nCore_NNT1 -> NNT1 rfN (0.000001) nCore_NNT2 -> NNT2 rfN (0.000001)

## B2. Rules for classifying fillers

Description	Example
S left filler	lfs -> A (0.0000001) lfs -> CSW (0.0000001) lfs -> D (0.0000001) lfs -> EX (0.0000001) lfs -> Fa (0.0000001) lfs -> Fa+ (0.0000001) lfs -> Fc (0.0000001)
S right filler	rfs -> A (0.0000001) rfs -> CSW (0.0000001) rfs -> D (0.0000001) rfs -> EX (0.0000001) rfs -> Fa (0.0000001) rfs -> Fa+ (0.0000001) rfs -> Fc (0.0000001)
NP left filler	lfn -> JJ (0.0000001) lfn -> CS (0.0000001) lfn -> FB (0.0000001) lfn -> ICS (0.0000001) lfn -> YI (0.0000001) lfn -> NP1 (0.0000001) lfn -> Tg (0.0000001) lfn -> JB (0.0000001)
NP right filler	rfn -> A (0.0000001) rfn -> Fa (0.0000001) rfn -> FB (0.0000001) rfn -> Fn (0.0000001) rfn -> Fr (0.0000001) rfn -> II (0.0000001) rfn -> Iu (0.0000001)
VP left filler	lfv -> CCB (0.0000001) lfv -> CS (0.0000001) lfv -> DA2 (0.0000001) lfv -> JJR (0.0000001) lfv -> MC (0.0000001) lfv -> PPH01 (0.0000001) lfv -> R (0.0000001)
VP right filler	rfv -> CS (0.0000001) rfv -> FA (0.0000001) rfv -> Fc (0.0000001) rfv -> Fc+ (0.0000001) rfv -> Fr (0.0000001) rfv -> DD2 (0.0000001) rfv -> P (0.0000001)

## C: The syntactic annotation for Susanne

Non-terminal	Description	Example
A	special "as" clause	A "as the deputy demanded"
D	determiner phrase	D@ "too much"
Fa	adverbial clause	Fa "As he passed through it"
Fa+		Fa+ "or not"
Fc	comparative clause	Fc "as it started to form"
Ff	fused relative	Ff "what you 've got there"
Fn	nominal clause	Fn "perhaps it was insane"
Fr	relative clause	Fr "which are hideous"
G	genitive phrase	G "the man's own"
I	interpolation	I "I feel"
lq	tag question	lq "didn't you"
lu	technical reference	lu "fig. 26"
J	adjective phrase	J "quiet"
J&		J& "daily and diurnal"
L	miscellaneous verbless clause	L "his left hand in front of him"
M	numeral phrase	M "a dozen"
M&		M& "five , six , or seven"
N	noun phrase	N "the three men"
N+		N+ "and oil"
N@		N@ "the only not red bag"
Ot	title (e.g. of book)	Ot "Othello"
P	prepositional phrase	P "by his yearning"
P@		P@ "to his office"
Q	quotation	Q "Oh , no"
R	adverb phrase	R "now"
S	main clause	S "There was no chance"
S@		S@ "do you understand that"
Tb	bare nonfinite clause	Tb "go of his arms"
Tf	"for-to" clause	Tf "for a cloud to move fast"
Tg	present participle clause	Tg "making sure he didn't see"
Ti	infinitival clause	Ti "to find his deeper self"
Tn	past participle clause	Tn "half shut"
V	verb group	V "was moving"
W	"with" clause	W "with water in their hats"
X	top level symbol	X any sentence
Suffix:		
+	subordinate with conj.	
-	Subordinate without conj.	
@	Appositional element	
&	Co-ordinate structure	